

DOCUMENT RESUME

ED 288 487

IR 012 878

AUTHOR Tempel, Michael; And Others
TITLE The ECCO Logo Project: Materials for Classroom Teachers and Teacher Trainers.
INSTITUTION International Council for Computers in Education, Eugene, Oreg.
REPORT NO ISBN-0-924667-41-9
PUB DATE [85]
NOTE 204p.
AVAILABLE FROM International Council for Computers in Education, University of Oregon, 1787 Agate Street, Eugene, OR 97403 (\$20.00; discounts on quantities available).
PUB TYPE Guides - Classroom Use - Materials (For Learner) (051) -- Guides - Classroom Use - Guides (For Teachers) (052)
EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS *Computer Assisted Instruction; Elementary Secondary Education; Instructional Materials; *Language Arts; Learning Activities; *Mathematics Instruction; Programing Languages; *Science Instruction; *Social Studies; Workshops
IDENTIFIERS *Educational Computer Consortium of Ohio; *LOGO Programing Language

ABSTRACT

In the fall of 1985, the Educational Computer Consortium of Ohio (ECCO) presented an extensive series of workshops on Logo. The workshops were divided into two categories: those for teacher-trainers and those for classroom teachers. This booklet presents materials developed by a core of five participants in the workshops for trainers using Logo that could be taken directly into the classroom. These materials are presented in two major sections: Logo in Language Arts and Social Studies, and Logo in Math and Science. Activities for teacher training are included. Each section begins with materials presented by Michael Tempel of Logo Computer Systems, Inc., in a series of workshops for teacher trainers. The second part of each section contains materials used in workshops for classroom teachers, which are designed to be directly usable by students in the classroom, and include student worksheets and teacher information sheets. Listings of Apple Logo II files for teacher training and student activities are appended. (RP)

* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *

About the authors:

Michael Tempel is Director of Educational Services for Logo Computer Systems, Inc. He has taught in the New York City public schools for 12 years and is well known nationwide for his presentations and teacher training workshops.

Brian Silverman was involved with the MIT Logo Projects and is currently Vice President for Research for Logo Computer Systems, Inc.

Ginny Frazier is a computer coordinator and teacher for the Hudson (Ohio) Public Schools. She has been involved with computers in education for many years.

Rick Hunyadi is a chemistry teacher at Wooster (Ohio) High School who enjoys computing and has taught computer science at the high school level.

Josie Drushal is the Elementary Computer Coordinator for the Wooster (Ohio) City Schools. She has been intimately involved in implementing Logo at the elementary level for the past several years.

Sharon Burrowes is the Computer Coordinator for the Wooster (Ohio) Public Schools and computer science teacher at Wooster High School. In addition to having taught Logo at the high school level for several years, she is actively involved in teacher training.

Ted Burrowes, a former mathematics and computer science teacher, is currently a systems analyst. He keeps his hand in the world of education through his work with Burrowes Software.

This book is published by the International Council for Computers in Education, the U. S. and international professional organization for computer-using educators. ICCE members are dedicated to improving education through appropriate use of computer-related technology. ICCE publishes *The Computing Teacher*, an international journal that focuses on instructional uses of computers at the precollege level.

QUANTITY	PRICE (U.S.\$)
1-4 copies	\$20.00 each
5-9 copies	\$18.00 each
10-99 copies	\$16.00 each
100+ copies	\$14.00 each

Please place your orders with:
International Council for Computers in Education
University of Oregon
1787 Agate Street
Eugene, Oregon 97403
503/686-4414

THE ECCO LOGO PROJECT:

MATERIALS FOR
CLASSROOM TEACHERS
AND
TEACHER TRAINERS

Edited by
Theodore C. Burrowes
and
Sharon K. Burrowes

TABLE OF CONTENTS

	PAGE
THE ECCO LOGO PROJECT	1
INTRODUCTION.....	2
TEACHER TRAINING:	
USING LOGO TO EXPLORE LANGUAGE	3
LOGO IN LANGUAGE ARTS AND SOCIAL STUDIES:	
Storytelling.....	12
Exploring Plurals.....	15
Exploring Sentence Structure.....	19
Generating Silly Stories.....	23
Creating a Student Newspaper.....	27
Changing Adjectives to Adverbs.....	31
Working with Words and Lists.....	35
Creating a Language.....	39
Using Compass Directions.....	43
Using a Database.....	48
Using Statistics to Analyze Data	52
TEACHER TRAINING:	
LOGO SCIENCE PROJECTS	56
FUZZY LOGO: LOGO IN MATH.....	66
LOGO IN MATH AND SCIENCE:	
Logo Arithmetic.....	77
Exploring Triangles.....	81
Exploring Polygons and Circles.....	84
Comparing Length	90
Exploring Perimeters.....	93
Estimating Distances and Angles.....	97
Coordinate Geometry.....	102
Exploring Sequences and Series.....	106
Exploring Limits.....	113
A Cartesian Graphing Microworld.....	119
The Turtle in Motion.....	127
Bouncing Turtles.....	134
Classification	138
APPENDIX 1:	
Listings Of Apple Logo II Files For Teacher Training Activities.....	143
APPENDIX 2:	
Listing Of Apple Logo Files For Student Activities.....	162

THE ECCO LOGO PROJECT

In the fall of 1985, the Educational Computer Consortium of Ohio (ECCO) embarked on a program of teacher training and materials development that was to last throughout the academic year. With the financial assistance of the Martha Holden Jennings Foundation, ECCO presented a rather extensive series of workshops on Logo.

The workshops presented during the year were divided into two categories: those for teacher trainers and those for classroom teachers. The workshops for teacher trainers were presented by Michael Tempel of Logo Computer Systems, Inc. These workshops were designed to give trainers a variety of experiences with Logo as well as provide general ideas for using Logo in the curriculum.

From among those trainers who attended Mr. Tempel's workshops, a core of five people developed materials for workshops for classroom teachers. Using the materials presented in the Tempel workshops, these five trainers developed specific materials that could be taken directly into the classroom and used to enhance the teaching of topics in language arts, social studies, science and math. These materials then served as the basis for workshops for the classroom teacher presented by this core of people during the year.

The year-long project culminated in a Logo Fair which included presentations by students and teachers and featured several outside speakers, including Seymour Papert. Some 300 to 400 teachers attended all or part of the Fair, which included half-day workshops and hour-long sessions on using Logo in the curriculum and the introduction of LogoWriter, the new version of Logo that had just become available from Logo Computer Systems, Inc. The excitement from the fair brought a large number of teachers into graduate workshops and courses during the following summer, bringing the project to its hoped for conclusion: from expert to trainer to teacher/learner to student.

INTRODUCTION

This booklet is divided into two sections: Logo in Language Arts and Social Studies, and Logo in Math and Science. The first part of each section contains materials presented by Michael Tempel in his workshops for teacher trainers. These workshops were done in Apple Logo II; files for these activities are found on the back of the enclosed disk. Listings of procedures used in these workshops that are not found in the written workshop materials, are found in Appendix 1 at the end of this booklet.

The second part of each section contains materials used in workshops for classroom teachers. These materials are designed to be directly usable by students in the classroom. Each activity is self-contained and includes a student worksheet and a teacher information sheet. The enclosed disk contains Apple Logo files for use with each activity. Listings (in Apple Logo) of all of the procedures used in each activity are found in Appendix 2 at the end of this booklet. In addition, Apple Logo II versions of the files are found on the back of the enclosed disk. (In order to keep them separated from the LCSII files also located there, they have been put in a subdirectory, so that if one of the activities asks you to LOAD "XYZ, you will instead need to LOAD "/LOGO/ICCE/XYZ.)

The files on the disk are locked to prevent accidentally erasing them. To unlock them use the DOS or PRODOS master disk.

Written By (in reverse alphabetical order):

Michael Tempel	Logo Computer Systems, Inc.
Brian Silverman	Logo Computer Systems, Inc.
Rick Hunyady	Wooster City Schools
Virginia Frazier	Hudson Public Schools
Josie Drushal	Wooster City Schools
Theodore Burrowes	Burrowes Software
Sharon Burrowes	Wooster City Schools

Final Layout:

David John Burrowes

Brown University, Class of 1990

USING LOGO TO
EXPLORE LANGUAGE

"Using Logo To Explore Language" and the associated files are © 1985 Logo Computer Systems, Inc. These are for personal use only, and are not to be copied or distributed without the express written permission of Logo Computer Systems, Inc.

Overview

Logo is well suited to the exploration of natural language. This is partly because the objects Logo works with are words and lists. A Logo word is a string of characters. So is an English word. In both Logo and written English, words are normally separated by spaces. A Logo list is an ordered collection of Logo objects. If these objects are words, we have a pretty good representation of an English phrase or sentence.

Since Logo procedures can manipulate Logo words and lists, they can work with English words, phrases, sentences, stories and poems.

Logo can also work with text files. A text file is just a sequence of characters saved on disk. The Logo editor may be used to write and edit a story or poem, which may be saved on a disk as a text file. Logo programs may then display this piece of writing, add some illustrations produced with Turtle graphics, and even manipulate and alter the text.

Logo also allows you freedom in naming objects and procedures. A procedure that draws a house may be called HOUSE. You can write a procedure called FASTER which speeds up the Turtle. This is taken for granted by Logo users, but is not a feature common to many other programming languages. It allows appropriate visual feedback to be evoked by English words.

Natural language activities in Logo fall into several general categories:

- Programs to explore the structure of words. These include pluralization, conjugation and spelling projects.
- Programs that examine how groups of words are assembled. Poetry and sentence generators are in this category.
- Conversation programs. These can be written by students. They are also written by teachers and used to prompt student writing.
- The Logo editor may be used to create and modify text. This allows one to write stories and poems that may be displayed under Logo program control. Graphics may be added for illustrations and Logo programs can be used to structure branching stories and adventure games.
- Programs that understand English words or phrases and give appropriate visual feedback. These projects are especially good for use with beginning readers.

Using the Disk of Sample Programs

The disk accompanying this booklet contains files of Logo programs and other text files used by some of the programs. Apple Logo II uses the ProDOS operating system. The volume name of the sample disk is /LOGO, which is the same name as the Apple Logo II program disk. If you're not familiar with the file system used by Apple Logo II, look at chapter 15 of the Reference Manual.

Word Structure

Often students are taught the rules of pluralization, conjugation or spelling and then asked to apply those rules by giving examples that illustrate each rule or exceptional case. Another approach is to ask the student to build the rules from specific cases. These rules may be embodied in a Logo procedure. When the procedure fails to behave properly it can be modified.

Let's take pluralization as an example. Most English words are pluralized by adding an "s." This can be written in Logo as:

```
TO PLURAL :WRD
  OP WORD :WRD "S
END
```

If you hand this procedure a word, you get it back with an "s" stuck on the end.

```
?PRINT PLURAL "CAT
CATS
```

```
?PRINT PLURAL "FOX
FOXs
```

The breakdown of PLURAL in the face of FOX can be fixed.

```
TO PLURAL :WRD
  IF EQUALP LAST2 :WRD "OX [OP WORD :WRD "ES]
  OP WORD :WRD "S
END
```

LAST2 is a procedure that outputs the last two characters of the word you give it. Look at the listing of the file PLURAL for details. That file also contains a more general solution to the FOXS bug.

- Try to develop this pluralization program as far as you can.

Note: Apple Logo II allows you to use upper and lower case text. It is "case insensitive" where procedures and names are concerned. This means that if you write a procedure called SQUARE, Logo will respond to SQUARE, square, Square and SqUaRe as equivalent. However, when you're examining text, a distinction is made between upper and lower case characters. Using the modified PLURAL procedure shown just above, PRINT PLURAL "FOX will give FOXES as expected, but PRINT PLURAL "fox will result in foxS. This is because the last two letters of "fox" are "ox," which is distinguished from "OX."

You can deal with this in several ways. For simplicity, just press the Caps Lock key and work only in upper case. You can also use the Logo operations UPPERCASE and LOWERCASE to make procedures that handle words in either case. UPPERCASE outputs the uppercase equivalent of the word you give it. For example

```
?PRINT UPPERCASE "fox
FOX
```

```
?PRINT UPPERCASE "Fox  
FOX
```

```
?PRINT UPPERCASE "FOX  
FOX
```

You could modify the PLURAL procedure using UPPERCASE

```
TO PLURAL :WRD  
IF EQUALP UPPERCASE LAST2 :WRD "OX [OP WORD :WRD "ES]  
OP WORD :WRD "S  
END
```

Conjugation presents a similar challenge. It's more interesting in French than in English. The file CONJUGATE contains examples in both languages that work with regular verbs. The French version only takes care of verbs with "er" endings.

- Elaborate the conjugation programs to cover irregular verbs. The French version should also accept regular verbs with "ir" and "re" endings.
- Write a program to generate the correct spelling of a word where there are alternative spellings of the same sound. For example, "cloud" and "clown" have the same vowel sound. What are the rules that determine when to use "ow" and when to use "ou"?

These sorts of programs are not routinely written by elementary school students. They are more appropriate as high school or college activities. However, it would be interesting to see how younger students would respond to such programs written by their teachers or older students. There could be a collaborative effort to test and refine the programs, with the more experienced Logo person providing the programming expertise as needed.

Sentence Structure

Rules of sentence structure, just like rules of pluralization and conjugation, may be expressed in Logo programs. SILLY is an example of this type of program. The procedure SILLY outputs a sentence made of words randomly picked from lists of articles, nouns, verbs and adverbs. The procedure PICK uses ITEM and RANDOM to do this. After loading the file type

```
?PRINT SILLY
```

several times. Type EDNS (for EDit NameS) to examine and change the lists of words that SILLY uses. The lists are named FAMILY, VERBS, NOUNS and ADVERBS.

PHRASES is similar to SILLY, but it displays each sentence in five steps. After loading the file, type the procedure name PHRASES to get it started.

POSTCARDS starts with a standardized format and generates examples. Type POSTCARD after loading the file.

You can modify the material that is used to produce the results. (Type EDNS to put the lists of words into the Logo Editor).

POSTCARDS also includes procedures that add to the lists of words and phrases. These are named LEARN.NEW.NAME, LEARN.NEW.PHRASE and LEARN.NEW.CLOSING. Type one of these procedure names. You will be prompted for an addition to the appropriate list. After adding your entry, run POSTCARD again several times. A card that includes your new item should show up sooner or later.

POETRY generates random rhymes from lists of nouns and rhyming words. Its method of selecting and associating words is somewhat more complex than the other examples. The noun list is used in rotation. After displaying a poem, the first noun is put at the end of the list. When the second poem is displayed, the next noun comes up first. The procedure ORDER.NOUNS puts the first noun at the end of the noun list and then outputs the new list.

The rhyming words are stored as a list of lists called RHYMES. When POETRY is run, a list of rhyming words is randomly picked from the larger list. The procedure SENTEN pairs nouns and rhymes until either the noun list or the list of rhyming words has run out. Then, pressing any key except the period causes POETRY to be called again, producing another verse.

- Develop a more complex sentence generator. To begin with, you could have a sentence assembled out of NOUN.PHRASE and VERB.PHRASE. NOUN.PHRASE could be made out of ARTICLE, ADJECTIVE and NOUN; VERB.PHRASE out of VERB and ADVERB.
- Try to make a Haiku generator. Each line requires a specific number of syllables. How can Logo keep track of this information?
- Start with some standardized form of written communication, such as an inter-office memo or a report card, and write a program to generate instances of the general category.

Conversation Programs

The file TALK contains several programs that carry on conversations. TALK1, like Weizenbaums's ELIZA, gives an appearance of being responsive but, in reality, is paying absolutely no attention to what its victim is saying. TALK5, on the other hand, exhibits some low-level intelligence.

- Pick a topic, such as a sport, kind of music or hobby, and write a conversation program that can carry on a discussion in the chosen domain.

Language Interpreters

Logo can be used to write programs that understand an English word or phrase and produce an appropriate response. Such programs are especially useful for children just learning to read. Simple words and phrases produce interesting visual responses. This encourages the use of written language.

There are two strategies for designing such programs. One is to write a Logo program which is a language interpreter. The files LANGUAGE1, LANGUAGE2 and LANGUAGE3 contain increasingly complex examples of this approach. In each of these files the procedure START gets things going.

The alternative is to create a collection of Logo procedures which work together to allow the user to get a response by typing an English sentence. LANGUAGE4 is an example of a program like this. It understands DRAW A PURPLE BALL or DRAW AN ORANGE SQUARE.

Since Logo syntax and English syntax are very different, there is a challenge in trying to get Logo to understand an English phrase. The interpreter approach bypasses Logo syntax in favor of the English syntax you can program Logo to understand.

The "Top Level" approach used by LANGUAGE4 has a harder time accommodating English syntax, but has the advantage of behaving like Logo as a whole. It can accept REPEAT 5 [FD 50 RT 144] as easily as DRAW A GREEN CIRCLE.

The larger memory of Apple Logo II allows much more comprehensive language interpreters to be developed.

- Write a program that can understand commands such as "draw 5 boxes" "show me six blue stars."
- Make a program that draws a person or face piece by piece as it is told to. Can you have it give responses such as "You already told me to draw the nose" to redundant commands?

Working With Text

The Logo Editor is just a text editor. In early microcomputer versions of Logo, this editor was intended only for writing procedures. You could enter the editor by typing EDIT followed by a procedure name or a list of procedure names. Leaving the editor caused its contents to be "evaluated": this means that Logo tries to make sense of each line as Logo code. It has the same effect as if you typed each line in at top level. For example, you could go into the editor and type

```
TO SQUARE
REPEAT 4 [FD 50 RT 90]
END
```

Leaving the editor would cause SQUARE to be defined just as if you had typed TO SQUARE at top level and then entered the next two lines in response to the ">" prompt. If you enter the editor and type

```
PRINT [THIS WAS WRITTEN IN THE LOGO EDITOR.]
```

then the message in brackets will be printed when you exit the editor. Logo evaluates the line, which is the command to PRINT something.

This evaluation of the contents of the Logo editor makes it useful for writing Logo code, but interferes with writing stories or other non-Logo text. If you wrote

```
Once upon a time ...
```

upon leaving the editor Logo would complain that it doesn't know how to Once, since it expects Logo code.

In Apple Logo II you have two alternative ways of using the editor. The command EDIT enters the editor from the Logo workspace and the contents of the editor are evaluated upon exiting. The command EDITFILE gets a text file directly from the disk (or creates a new one if the named file doesn't already exist). Upon leaving the editor, the modified text is written back to the disk. It is not evaluated. For example:

```
EDITFILE "STORY
```

would put the contents of the text file STORY in the editor, or create a file named STORY. Leaving the editor would put the new version of STORY on the disk.

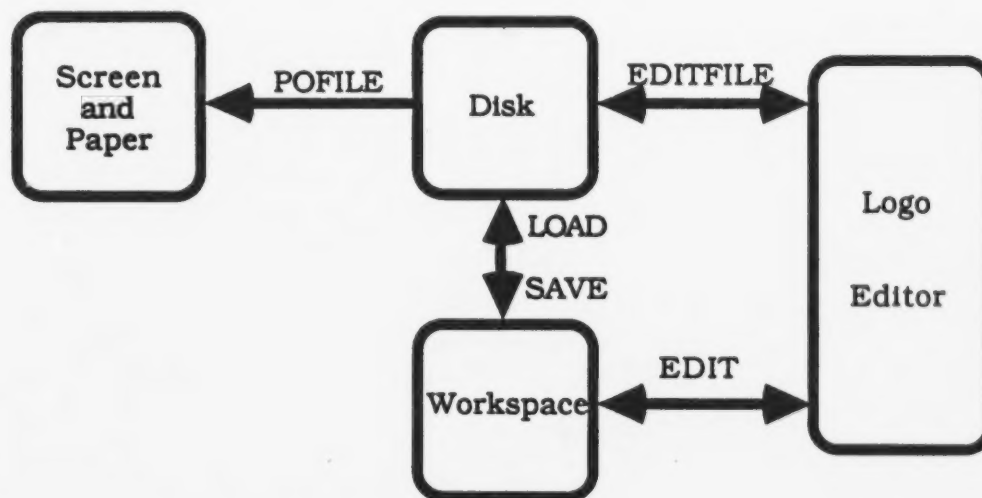
```
POFILE "STORY
```

will display the contents of the file on the screen. If you want it on paper

```
DRIBBLE 1  
POFILE "STORY  
NODRIBBLE
```

will do the trick, assuming your printer is connected to slot 1 of the Apple.

This diagram summarizes the above discussion:



Disk files may also be read a character, a word or a list at a time using READCHAR, READWORD and READLIST in conjunction with SETREAD. Files may be written to using PRINT and TYPE along with SETWRITE. There's more information about this below, under "Altering Text."

The file STORIES has three versions of an illustrated story that alternately displays text files using POFILE and familiar Turtle graphics designs. The text files CHAPTER1 CHAPTER2, etc., were created using EDITFILE. They form the text of the stories. Procedures such as SQUARE, HOUSE and VILLAGE take care of the graphics.

STORY is the simplest version. It just displays alternate text and graphics using the WAIT command to give you time to read each piece.

STORY2 uses the procedure IGNORE to cause an indefinite pause in the action. Pressing any key continues the procedure. STORY2 also asks you how many houses you want in your village and builds a development of the requested density.

STORY3 includes a branching option which allows you to select an ending for the story.

- Create a branching story or adventure game along the lines of STORY3.
- Instead of using procedures to draw all your graphics, create a drawing, save it with SAVEPIC and have your STORY program load it at the appropriate time using LOADPIC.

Altering Text

Logo procedures may perform many familiar text editing tasks. The file TOOLS includes some examples.

DELETE removes the first occurrence of an item from a list. For example:

```
?PRINT DELETE "APPLES [PEARS APPLES GRAPES]
PEARS GRAPES
?PRINT DELETE "A [A B C B A]
B C B A
```

DELETE2 removes all occurrences.

```
?PRINT DELETE2 "A [A B C B A]
B C B
```

REPLACE substitutes one item for another.

```
?PRINT REPLACE "IBM "APPLE [I LIKE MY IBM COMPUTER]
I LIKE MY APPLE COMPUTER
```

REPLACE2 replaces all occurrences of the first item with the second.

The procedure ALTER reads a text file from disk and displays a new version with a word substitution. The file MOON contains the opening lines of the book Goodnight Moon by Margaret Wise Brown. First type

```
?POFILE "MOON
In the great green room
There was a telephone ...
```

to see the original text. Then try

```
?ALTER "MOON "green "blue
In the great blue room
There was ...
```

The word green is changed to blue in the first line. ALTER2 takes a list of pairs of words and performs all the substitutions.

```
?ALTER2 "MOON [[green blue][telephone computer]]
In the great blue room
There was a computer
And a ...
```

ALTER3 is the same as ALTER2 but it writes the modified text to a new disk file instead of displaying it on the screen. After running ALTER3, use POFILE to see the result.

```
?ALTER3 "MOON "NEW.MOON [[green blue] [telephone computer]]
?POFILE "NEW.MOON
In the great blue room
There was a computer
And a ...
```

- Use a REPLACE procedure in conjunction with PLURAL and CONJUGATE to create a procedure that takes something like [THE DOG RUNS] as input and outputs [THE DOGS RUN]
- Create a procedure that capitalizes selected words
- Write a program that alphabetizes a list of words. (Look at the primitive procedure BEFOREP.)
- Write a Mad Libs program.
- Write a random poetry or sentence generator that works on text files using a procedure such as ALTER.

LOGO IN
LANGUAGE ARTS
AND
SOCIAL STUDIES

STORYTELLING

Directions for Large Group Instruction

Be familiar with the story Aranea: The story about a Spider by Jenny Wagner (Bradbury Press, 1975).

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "STORYTELLING

Now, tell your students that they are going to hear a story about a spider named Aranea. Indicate that you will use some computer pictures to help tell the story.

For grades K-2:

"Think of other pictures you could add to the story besides the ones on the computer screen."
(Suggestions: a spider web, lightning, a lilac bush, Aranea's leaf. Some older students with experience using Turtle graphics and writing Logo procedures could be asked to make these pictures.)

For grades 2-6 (programming students):

"As you are listening, try to think of other pictures you would like to add to the story."
Then type

SPIDER

and press RETURN and the Logo Turtle will draw Aranea, the spider.

Continue telling the story.
When Aranea gets caught in the rain type

RAIN

and press RETURN.

Followup Discussion and Projects with Programming Students:

Assign groups of students to work on certain graphics for the story. Tell the story again (perhaps share it with another class) when the new graphics are finished.

Choose a favorite story and program Logo graphics to go with the story. Share it with your class or another class.

Teacher Information Sheet for Storytelling

Objectives:

- (1) Students will listen to a story to discover the main characters or events.
- (2) Students interested in programming will create Turtle procedures to accompany the program.
- (3) Nonprogramming students will be introduced to Turtle graphics and enjoy the illustrations for the story on the computer.

Grade level:

Students in grades K-2 can listen to the story and enjoy the Logo pictures that accompany the story.
Students in grades 2-6 familiar with Turtle graphics, writing procedures, and saving files to a disk can use the Turtle to make additional characters or events for this story or other stories.

The idea for this activity was adapted from a suggestion by Candy Colborn, Media Specialist, Cottonwood Creek Elementary School, Englewood, Colorado.

EXPLORING PLURALS

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "PLURALS

You are going to explore making plurals of words.
To begin, type

PRINT PLURAL "DOG

Logo types _____

Try

PRINT PLURAL "HOUSE

Logo types _____

Now you try some words. Be sure to put " in front of your word.

You typed:

Logo printed:

Did Logo always give the right answer? _____ Circle any wrong answers.

Now try typing

PRINT PLURAL "BOX

Logo types _____

'Boxs' is not right. It should be 'boxes'. Can you think of other words like box? List several:

Is there a rule for these words? What is it?

Are there other types of words that don't work? List groups of words and see if you can figure out a rule for fixing the plurals.

Group 1: _____

Rule: _____

Group 2: _____

Rule: _____

Group 3: _____

Rule: _____

=====

For Logo Programmers:

Type

ED "PLURAL

Change the procedure so that it looks like this:

```
TO PLURAL :WORD.IN
  IF LAST :WORD.IN = "X [OUTPUT WORD :WORD.IN "ES]
  OUTPUT WORD :WORD.IN "S
END
```

Now try typing

PRINT PLURAL "BOX

Does it work correctly? _____

Here is a brief explanation of some things you may not know about Logo:

LAST outputs the last letter in a word that follows it .
WORD "sticks together" the two things that follow it into a single word.
OUTPUT sends results out of the procedure and stops the procedure from running anymore.

In addition, there are two Logo procedures, LAST2 and LAST3, available in the PLURALS workspace (and listed in Appendix 2.) Both need a word as input.

LAST2 reports the last two characters of the word.
LAST3 reports the last three characters.

Thus, if you type

PRINT LAST2 "BOY

Logo prints "OY."

Now, can you change PLURAL so that it works correctly for some of the word groups you found earlier?

Write your improved version(s) of PLURAL here:

Teacher Information Sheet for Exploring Plurals

Objectives:

- (1) Students will explore how plurals are formed.
- (2) Students will learn to use the Logo primitives WORD, LAST and OUTPUT.
- (3) Students will be encouraged to expand the program provided and thereby learn more about both pluralization and Logo.

Grade level:

This project is applicable to almost any grade level. Very young students would only use the Logo procedure given; older and more sophisticated students would be expected to expand the given procedure.

Discussion:

This unit should be used when students are working with pluralization. Encourage experimentation. Let students find out what words work and what words don't. Class discussions may well lead to various pluralization exceptions and rules. Some students will collect lists of similar words; others will seek rules. Encourage a diversity of strategies rather than focusing on just one.

If students are more experienced with Logo, they may wish to expand the program. For example, suppose they discover that words ending in X have to have an ES added. Then they would need a statement such as:

```
IF LAST :WORD.IN = "X [PRINT WORD :WORD.IN "ES]
```

Or, perhaps they might choose to use the tool procedures provided: LAST2 and LAST3. For example, if they discover that words with certain two letter endings need ES added, they might use

```
IF MEMBERP (LAST2 :WORD.IN) [SH CH OX AX SS]  
[OUTPUT WORD :WORD.IN "ES]
```

These statements would have to come before the OUTPUT statement in the original procedure. As students add new checks, then the order of the IF statements becomes more important.

Common mistakes in adding IF statements are likely to be:

- (1) leaving a space between : and WORD.IN
- (2) using parentheses () instead of square brackets []
- (3) omitting square brackets (in LCSI versions of Logo)
- (4) leaving a space between " and X or " and ES

EXPLORING SENTENCE STRUCTURE

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "SENTENCES

For this activity, you are going to explore the way sentences are put together. Begin by typing

NOUN VERB .

(Be sure to leave a space before the period.)

What does Logo print? _____

Try typing **NOUN VERB .** several times. You can have Logo repeat the same thing you just typed by holding down the Control key and pressing the Y key, and then pressing Return.

Describe what happens.

Now, add an article to the sentence.
Type

ARTICLE NOUN VERB .

What does Logo print? _____

Try this several times.

What do you think would happen if you changed the words that Logo is using to create the sentences?
You can experiment by typing

EDNS

Among the information on the screen, you will then see

MAKE "NOUNS [CAT BIRD HORSE STUDENT]

Move the cursor so that it is on top of the] after the word STUDENT. Press the space bar, and then type some more nouns. Don't press Return. It may help you to know the following key combinations to move around:

<- deletes characters
-> moves to the right
Control/B moves to the left (Backward)
Control/P moves up a line (Previous line)
Control/N moves down a line (Next line)

When you are done adding nouns, press Control/C.

Now have Logo create some more sentences. Can you describe the kind of nouns that make sense in these sentences? Are they names of living things or objects or proper nouns, etc.?

Next, type

EDNS

and add some verbs. Have Logo create some sentences. Can you describe the kind of verbs that make sense? Are they action verbs or descriptive verbs, etc.?

Next type

ADVERB .

Do you see that this prints an adverb? Experiment with putting an adverb into your sentence. What word order makes sense? Is there more than one possibility?

You can change the adverbs by typing

EDNS

again. What kind of adverbs work well? Is there a pattern to be found in the type of adverbs or some other characteristic in common? Can you use any adverb you want?

Using **EDNS** you can change any or all of the words. Experiment with different groups of words. Write any observations that you make here:

For Logo programmers:

Try adding more parts of speech. If you type

ED [NOUN VERB ARTICLE ADVERB]

you will see that all of the procedures follow the same pattern. You will also have to add a list of words to use. You can do this by using the **MAKE** statement. For example,

MAKE "ADJECTIVES [PRETTY HUNGRY TIRED]

Try adding adjectives, pronouns, interjections, and so forth. You might even try adding conjunctions and creating compound sentences.

Write the procedures you develop here:

Experiment with the new procedures you added. What discoveries did you make about word order.?

Did you look at the procedure **PICK** that is used in all of the parts of speech procedures? It uses a number of primitives that you may not know:

OUTPUT sends information out of the procedure back to the calling procedure
ITEM takes a number and a list as inputs. It returns the 'number-th' item in the list.
COUNT takes a list as input and returns the number of elements in the list.

Experiment with **PICK** to see how it works.

Teacher Information Sheet for Exploring Sentence Structure

Objectives:

- (1) Students will review several parts of speech.
- (2) Students will have an opportunity to experiment with word order and see how sentences must be put together.
- (3) Students interested in programming will learn the MAKE statement and see how the PICK procedure is used to randomly select from a list.

Grade level:

This activity would be most appropriate for students at the junior high level or higher, although a simplified version could be used with younger children who are just learning the names of the parts of speech.

Discussion:

This activity could be used to reinforce the names of the parts of speech for younger children. For older or more sophisticated students, exploration of possible word order or examination of subcategories with a part of speech (e.g. transitive and intransitive verbs) is more appropriate.

If you have some programmers who can extend this microworld for you, then a richer exploration can take place. The older the student, the more complex you can make this exploration.

You should recognize that this activity allows students to "fiddle with" parts of speech in an unusual way. It is more common to write the sentence and then identify the part of speech for each word. This reverses the process and may enlighten students as to *why* we categorize words as particular parts of speech.

Students may become fascinated with finding words that work as several parts of speech or even make unexpected discoveries. Be prepared for discussion of their discoveries.

Common problems:

- (1) Students will not leave spaces when adding words or will omit brackets.
- (2) Students will forget the period after asking for a sentence. This will not cause a problem except that the *next* thing typed will appear on the same line as the sentence created.
- (3) Students who work on the programming section may not be careful about following the pattern used to name procedures and variables. Notice that here, the procedure names are always singular but the corresponding variable names are plural.

GENERATING SILLY STORIES

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "SILLY.STORIES

You are going to be asked to give information that will be used by the computer to print a story on the screen.
Type

STORY1

NAME A GIRL.

appears on the screen

What girl did you name? _____

Write your answers to the other questions that appear in the blanks:

NAME A PIECE OF FURNITURE. _____

NAME A KIND OF FOOD. _____

NAME ANOTHER FOOD. _____

NAME AN ANIMAL. _____

NAME A VERB IN THE PAST TENSE. _____

What happens next? _____

Does anything in the story look familiar? _____

Type **STORY1** again and give six different answers.

What happens this time? _____

Try a new story.
Type

STORY2

NAME A NUMBER. _____

NAME THE PLURAL OF AN ANIMAL. _____

Example: FOXES

NAME A VERB. _____

NAME A FEMALE IN A FAMILY. _____

Example: SISTER

NAME A PLURAL FOR A PART OF THE BODY. _____

Example: ARMS

NAME A SHARP OBJECT. _____

What happens? _____

Type **STORY2** again and answer the 6 statements with new words.

=====

For Logo programmers:

You can create your own silly story.

1. Write a rhyme.
2. Underline one or two words in each line that could be changed to make a silly story.
3. Examine the procedures **STORY1** and **STORY2** to see how they work.
4. List each word you underlined below and the Logo statement you would use to get words to put into the poem.

WORD	LOGO STATEMENT
_____	NAME _____ READLIST
_____	NAME _____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

5. Now put these together as a procedure to create a silly poem that works like STORY1 and STORY2.
Write your procedure here:

Now for a challenge. Can you write a silly story procedure that has some of the funny words at the end of the line and still be sure that the poem rhymes?

Teacher Information Sheet for Generating Silly Stories

Objectives:

- (1) The student will explore parts of speech in context (nouns, adjectives, verbs).
- (2) For programming students:
The student will explore the use of NAME and READLIST.
The student will discover the use of SENTENCE.

Grade level:

This would be appropriate for all elementary students. Beginning readers would enjoy giving their responses to an adult or older student and having their silly story read to them.

Discussion:

No previous experience with Logo is needed for the first part of this activity. Students with programming experience in writing and editing procedures could create their own stories using the procedures STORY1 and STORY2 as models.

Information needed for programming students:

SENTENCE is an operation that needs two inputs that are either words or lists. It puts its inputs into a single list. Notice that () are needed around SENTENCE when there are more than two words or lists. For example:

```
PRINT SENTENCE [THE DOG] :VERB
```

does not require parentheses, while

```
PRINT (SENTENCE [THE DOG] :VERB [TO THE HOUSE.])
```

does.

CREATING A STUDENT NEWSPAPER

Student Worksheet

- (1) Write below the important information and facts for your news story.

- (2) After reading the information above, fill in the correct information below:

WHO _____

WHAT _____

WHEN _____

WHERE _____

- (3) Write a short news story including WHO, WHAT, WHEN and WHERE within the first two sentences.

Teacher Information Sheet for Creating A Student Newspaper

Objectives:

- (1) The student will learn to include WHO, WHAT, WHEN and WHERE in the opening of a newspaper article.
- (2) The student will learn to use PRINT in Logo to format the text on the screen.

Grade level:

Students will need to feel comfortable writing procedures and using the Editor to do this activity. The recommended grade levels are 4-8.

Discussion:

This activity could be used for any project in which text needs to appear on the screen (short reports in science, social studies, messages to students for holidays, etc.).

The teacher will need beginning-level competence in Logo. The NEWS.EXAMPLE Logo file on disk and in Appendix 2 is a short example of a two-story newspaper.

To make the screen easier to read, a blank line is inserted after each line of text. This is accomplished by: PRINT []

At the end of each screen of text, the following procedures allow the student to press RETURN when he/she finishes reading the screen. The screen clears (CLEARTEXT) and is ready for the next page of text (or Turtle graphics). These procedures serve as a "page turner."

```
TO NEXTSCREEN  
PRINT [PRESS RETURN TO CONTINUE]  
IGNORE RC  
CLEARTEXT  
END
```

```
TO IGNORE :THING  
END
```

Your lesson plan for this activity might go as follows:

- (1) Sample articles from newspapers should be used to show students how to place the important information WHO, WHAT, WHEN and WHERE at the beginning of an article. The students should underline these important facts on several articles.
- (2) The teacher loads Logo into the computer and then types:

```
LOAD "NEWS
```

followed by

```
NEWS
```

and the first news article appears on the screen. The teacher should have the students notice the following:

- All important information is in the first sentence or two (WHO, WHAT, WHEN, WHERE).

- There is only room for 39 characters (letters, spaces, numbers or punctuation) on each line of the screen.
 - The screen is easier to read when there is a blank line between each line of text.
 - The title of the story is two lines together. Then there are two blank lines to separate the title from the story. The story can only be 10 lines on each screen.
- (3) Students are divided into groups of three or four and are then assigned a "news story" and told to gather the necessary information using the Student Worksheet.
- (4) After the Student Worksheet is completed, the teacher gives each group a name for their procedure, and students enter their information using Logo.

```
TO NEWS1...
TO NEWS2...
TO NEWS3...
TO NEWS4...etc.
```

- (5) When the students have finished entering their news articles, the teacher or a capable student creates a superprocedure.

```
TO NEWSPAPER
NEWS1
NEWS2
NEWS3
NEWS4...etc.
END
```

- (6) This is then saved by typing

```
SAVE "NEWSPAPER
```

- (7) Additional ideas for more capable students:

A HEADER (using Turtle graphics), which gives a title to the newspaper, could be added. The procedure NEXTSCREEN would need to be used just before END in the HEADER procedure. Then HEADER would be the first procedure inside the superprocedure NEWSPAPER.

Additional graphics (illustrations for the news stories) could be added to the superprocedure. The superprocedure NEWSPAPER would then be:

```
TO NEWSPAPER
HEADER
NEWS1
PIC1
NEWS2
PIC2
NEWS3...etc.
```

CHANGING ADJECTIVES TO ADVERBS

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "ADVERBS

You are going to experiment with changing adjectives to adverbs. To begin, type

PRINT ADVERB1 "LOUD

Logo responds _____

Try

PRINT ADVERB1 "SOFT

Logo responds _____

Now try entering some adjectives of your own choice. Be sure to put " in front of each adjective.

You typed:

Logo responded:

Did Logo always give the right answer? _____

Circle any wrong answers.

Try typing

PRINT ADVERB1 "PRETTY

Logo responds _____

'Prettyly' is not right. It should be 'prettily.' Can you think of other words like pretty? List several:

Is there a rule for these words? What is it?

Try typing

PRINT ADVERB2 "PRETTY

Does ADVERB2 work for all the words in your list above?
List any words that still do not work right.

Next, try typing

PRINT ADVERB2 "GAY

PRINT ADVERB2 "GREY

What is wrong with the answers you get?

Now try

PRINT ADVERB3 "GAY

PRINT ADVERB3 "GREY

What is the difference between the rule for making adverbs from words like 'pretty' and the the rule for words like 'grey'?

Can you find any adjectives that won't work with ADVERB3? List them.

Try typing

PRINT ADVERB3 "TERRIBLE

PRINT ADVERB3 "GENTLE

What rule is needed to make adverbs from these adjectives?

Now try

PRINT ADVERB4 "TERRIBLE

PRINT ADVERB4 "GENTLE

Can you find any adjectives that won't work with ADVERB4? List them.

Try

PRINT ADVERB4 "GOOD

PRINT ADVERB4 "FAST

Now try

PRINT ADVERB5 "GOOD

PRINT ADVERB5 "FAST

List any adjectives that you can find that won't work with ADVERB5. Are they "exceptions" or are there rules that can be followed to make the adverbs?

For Logo programmers:

Examine each of the procedures that are in the workspace by typing

ED [ADVERB1 ADVERB2 ADVERB3 ADVERB4 ADVERB5]

Study how each procedure has been changed from the one preceding it.
Here is a brief explanation of some things you may not know about Logo:

AND works like the word "and" in English except that it must go in front, so instead of saying "bread and milk", you would say "and bread milk" in Logo.

BUTLAST gives everything but the last letter in a word that follows it.

LAST gives the last letter in a word that follows it.

MEMBERP checks to see if the first object following it is in the second object following it.

OUTPUT sends results out of the procedure and stops the procedure from running any more.

WORD "sticks together" the two things that follow it into a single word.

Now can you modify ADVERB5 to handle any adjectives that will not work with ADVERB5? For example, the words "economic" and "enthusiastic" will not work with ADVERB5. Write your new procedure here:

Teacher Information Sheet for Changing Adjectives to Adverbs

Objectives:

- (1) Students will explore how adverbs are formed from adjectives.
- (2) Students with some knowledge of programming will be encouraged to expand the program provided and thereby learn more about both the formation of adverbs and Logo.

Grade level:

This project is probably most applicable to students at the junior high school level and higher. Students with little Logo programming knowledge can focus on the formation of adverbs; students knowing how to program in Logo can work on expanding the procedures given in the file.

Discussion:

When using this project, encourage experimentation. Let students find out what words work and what words don't. Class discussions may be productive to clarify the differences between adjectives and adverbs as well as to find exceptions for each procedure.

The teacher need not be familiar with programming to use this activity. However, if either the teacher or students wish to pursue the programming activities, then some knowledge of Logo list processing is needed.

In case you are not a Logo programmer, here is a summary of what each procedure does:

ADVERB1 adds LY to any word entered.

ADVERB2 removes the Y and adds ILY if the word entered ends in Y; adds LY otherwise.

ADVERB3 removes the Y and adds ILY if the word entered ends in Y and the next to last letter is not a vowel; adds LY otherwise

ADVERB4 removes the Y and adds ILY if the word entered ends in Y and the next to last letter is not a vowel; removes the E and adds Y if the word ends in LE; adds LY otherwise.

ADVERB5 deals with the exceptions GOOD, HARD, FAST, EARLY and LATE; removes the Y and adds ILY if the word entered ends in Y and the next to last letter is not a vowel; removes the E and adds Y if the word ends in LE; adds LY otherwise.

If students are more experienced with Logo, they may wish to expand ADVERB5 to include any exceptions found. Encourage them to look for patterns rather than to include long lists of statements such as

```
IF :WORD.IN = "<some word here> [OUTPUT "<correct adverb>]
```

Common mistakes in Logo programming are likely to be:

- (1) leaving a space between : and WORD.IN
- (2) omitting square brackets after IF statements
- (3) mismatched or omitted parentheses
- (4) not putting AND before two conditions

WORKING WITH WORDS AND LISTS

Student Worksheet #1

PRINT is used to put messages on the screen. You must be careful to use the correct punctuation with the words that follow the PRINT command.

Type these statements:	Write what happens here:
PRINT "LOGO	
PRINT LOGO	
PRINT [LOGO]	
PRINT "I "LIKE "LOGO	
PRINT [I LIKE LOGO]	
PRINT I LIKE LOGO	
PRINT [I] [LIKE] [LOGO]	
PRINT ["I LIKE LOGO]	

What punctuation do you use to print a single word? _____

What punctuation do you use to print more than one word? _____

How many inputs does PRINT need? _____

What happens when PRINT has more than one input? _____

Student Worksheet #2

A Logo word is a collection of characters. A Logo list is a collections of words and/or lists. A Logo number is a special kind of word.

Notice how numbers differ from other Logo words.

Type these statements:	Write what happens here:
PRINT "HELLO	
PRINT "12345	
PRINT HELLO	
PRINT 12345	
PRINT [12345]	
PRINT "3 + "4	
PRINT 3 + 4	
PRINT [3 + 4]	

What punctuation do you use to print the answer to a math problem? _____

What punctuation do you use to print a math problem? _____

What is the difference between numbers and other Logo words? _____

Student Worksheet #3

Logo has a number of reporters (also called operations) for working with words and lists.

Type these statements:	Write what happens here:
PRINT FIRST "TURTLE	
PRINT FIRST [LOGO IS FUN]	
PRINT BUTFIRST "TURTLE	
PRINT BUTFIRST [LOGO IS FUN]	
PRINT LAST "TURTLE	
PRINT LAST [LOGO IS FUN]	
PRINT BUTLAST "TURTLE	
PRINT BUTLAST [LOGO IS FUN]	

What does FIRST do with a word as input? _____

What does FIRST do with a list as input? _____

What does BUTFIRST do with a word as input? _____

What does BUTFIRST do with a list as input? _____

Describe how LAST and BUTLAST work: _____

Teacher Information Sheet for Working With Words and Lists

Objectives:

- (1) The students will become familiar with the punctuation used to print words and lists.
- (2) The students will examine how printing numbers differs from other words.
- (3) The students will become familiar with the word and list commands: FIRST, LAST, BUTFIRST and BUTLAST.

Grade level:

This would be appropriate for grades 3-8 to show students that Logo can also work with words and is more than Turtle graphics.

Discussion:

No previous experience with Logo is needed for this activity. Since most students begin Logo with Turtle graphics, this activity will probably be used with students who have some experience with Logo. The teacher should be familiar with Logo's list processing capabilities.

The teacher should be sure that students are using the textscreen so that they can see more than four lines of their work. Students can then complete the worksheets either individually or with a partner.

The first worksheet focuses on the importance of punctuation and spaces when printing words and lists. The second examines the use of numbers with PRINT. The third explores the use of the Logo primitives FIRST, BUTFIRST, LAST and BUTLAST.

Time should be spent with groups of students who have completed the worksheets to check the sheets and be sure each student understands the ideas presented on each sheet. Encourage curious students to experiment further in ways of their own choosing.

The most common errors are likely to be incorrect use of the space. Watch for spaces between the " symbol and the word following, or omitting spaces that are shown on the worksheet.

Note: This activity has no file on the disk or printed procedures in the appendix because it uses no Logo procedures.

CREATING A LANGUAGE

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "LANGUAGE

This activity assumes that you can use the Logo editor. If you do not know how, perhaps you can work with another student who knows how to use the editor.

Type

SQUARE

Logo draws a square. That is the way we usually use Logo. Now type

DRAW SQUARE

Logo still draws a square. What does DRAW do? Type

ED "DRAW

and see what it does. Now leave the editor and type

DRAW A TRIANGLE

What does A do? Type

ED "A

to see. Do you see why the sentence you typed does what is expected?

This activity involves getting Logo to understand increasingly complex sentences.

Suppose you want to have Logo draw colored shapes. You might want to say

DRAW A BLUE SQUARE

(The workspace already contains SQUARE, TRIANGLE, and RECTANGLE) Can you write procedures for some colors that will work as shown above? Write them here and type them into your Logo workspace.

TO BLUE

TO GREEN

END

END

TO ORANGE

TO VIOLET

END

END

Now, type

DRAW A VIOLET SQUARE

and

DRAW A BLUE TRIANGLE

What do you need to add to the workspace to be able to type

DRAW AN ORANGE RECTANGLE

instead of

DRAW A ORANGE RECTANGLE

Now try typing

DRAW A RECTANGLE

What color is the rectangle? Do you want it to be the same color as the last shape? How can you fix this?
Write your modifications and/or additions here:

How about size. Can you write procedures so that you can type

DRAW A BIG RECTANGLE

or

DRAW A SMALL SQUARE

Write your modifications/additions here:

Now type

DRAW A TRIANGLE

Is the triangle big or small? Shouldn't it return to the original size?
Write here the changes you need to make for this to work correctly.

Here are some other ideas for you to try to add to your language:

BIGGER, SMALLER

ABOVE, BELOW and BESIDE

INSIDE and OUTSIDE

RIGHT and LEFT (You'll need to use something like RRIGHT because RIGHT is a primitive.)

AND and OR (These are primitives, too...and you may find that using EITHER...OR works better!)

TOUCHING

ON TOP OF

SAME SIZE AS

SAME COLOR AS

...use your imagination!!

Be sure to save your successes!

Teacher Information Sheet for Creating a Language

Objectives:

- (1) Students will gain experience programming in Logo.
- (2) Students will gain an understanding of what makes a sentence work within a limited environment.

Grade level:

This is most appropriate for students who are comfortable with the concept of a variable and will probably work best with students who are middle school age or older. The older the student and/or the more Logo programming s/he knows, the more sophisticated this activity can become.

Discussion:

Since this is primarily a programming activity, it may not be appropriate for all students. The teacher should be comfortable programming in Logo but does not need to have "all the answers." This activity merely needs a good facilitator. However, if you pair a student who can program with one who cannot, even more ideas may be generated. You may want to have students demonstrate successful procedures.

Classroom discussion will probably center on such questions as why some words simply do nothing and are still used in our language. If students get far enough, they will discover that some concepts can only be translated backwards, e.g., for shape A to be above shape B, shape B must be below shape A. You should be prepared for a lot of questions about the structure of language.

Possible problems:

The problems that arise will be those that arise in any programming activity. This will involve the need to help debug procedures. Often a few good students can serve in this role.

Encourage students to think modularly. For example, as the language created gets more complex, more and more will probably be added to each shape procedure. Suggest to students that all of these statements can be put in a separate procedure that is called from each shape procedure.

Don't let students get bogged down. If they are making no progress on one idea, suggest that they try another. Encourage sharing of ideas among groups of students.

USING COMPASS DIRECTIONS

Student Worksheet

Load Logo into the computer.
Put the files disk into the disk drive.
Type

LOAD "COMPASS

Type

CS

With masking tape, put Map #1 on the screen with the triangle over the Turtle.

Using the directions attached to your computer, help the Turtle find her way to the Turtle pond.

When you have finished the first map, remove it from the screen and attach Map #2. Help the Turtle find her friend who is asleep.

When you have finished the second map, remove it from the screen and attach Map #3. This time help the Turtle find the lost dog.

Tape Map #1 on the screen again. Try to give the Turtle all the directions to go to town at the same time. Give all the directions (separated by a space) and then press Return. For example:

S W N N N S.W N N N Return

Try to type all of the directions at once for Map #2 and Map #3.

Ask the teacher for a transparency and design your own map for you or a friend to guide the Turtle through.

Teacher Information Sheet for Using Compass Directions

Objectives:

- (1) The students will reinforce their directional map skills.
- (2) The students will create mazes for other students to use to practice map skills.

Grade level:

These activities are appropriate for intermediate grade students and require no programming skills.

Discussion:

Be sure to have transparencies of the three maps (make copies from the maps on the following three pages) beside each computer before you pass out student worksheets.

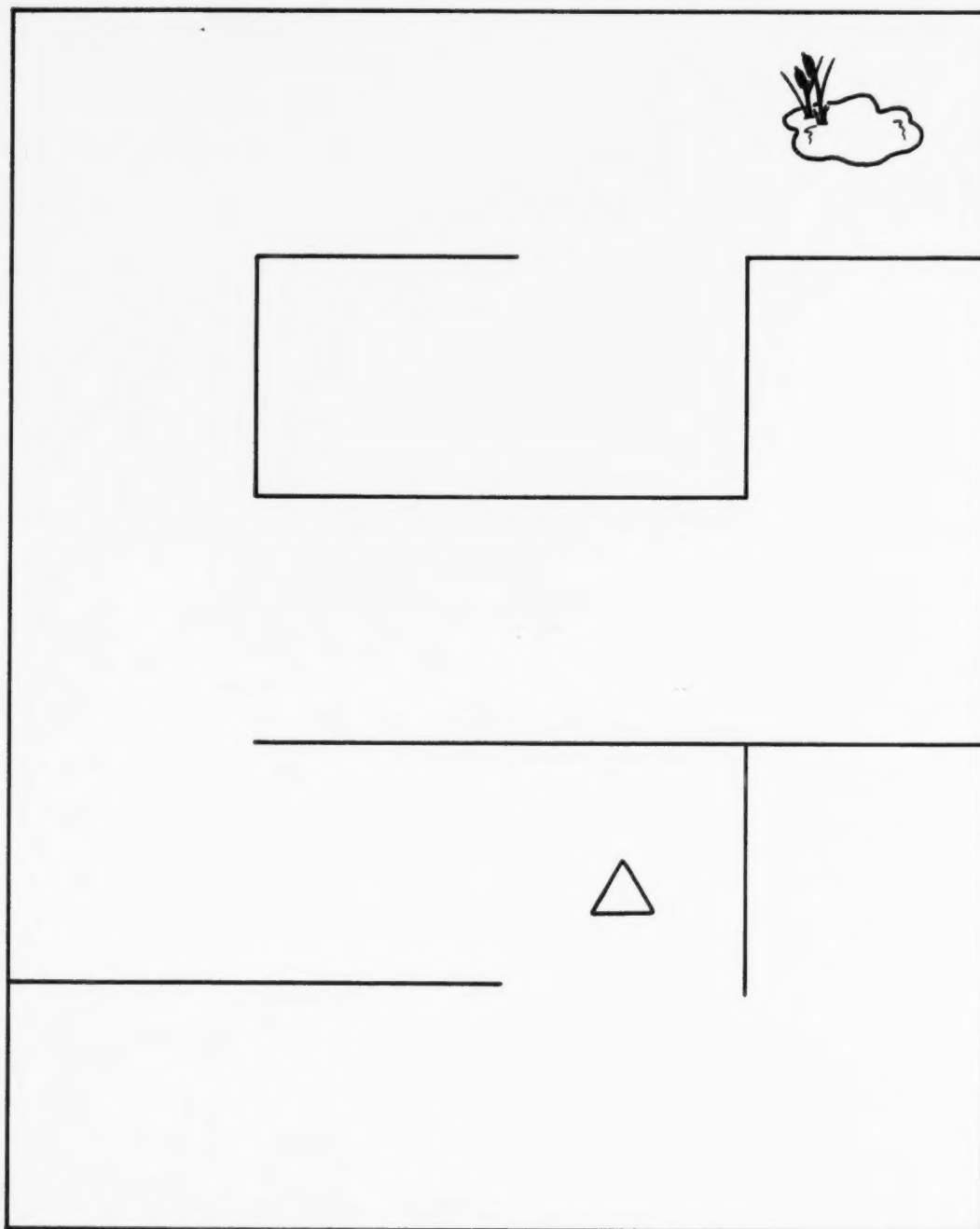
Attach a card with these directions to each computer:

**TYPE THE CORRECT DIRECTION FOR THE
TURTLE TO MOVE
AND THEN PRESS RETURN.**

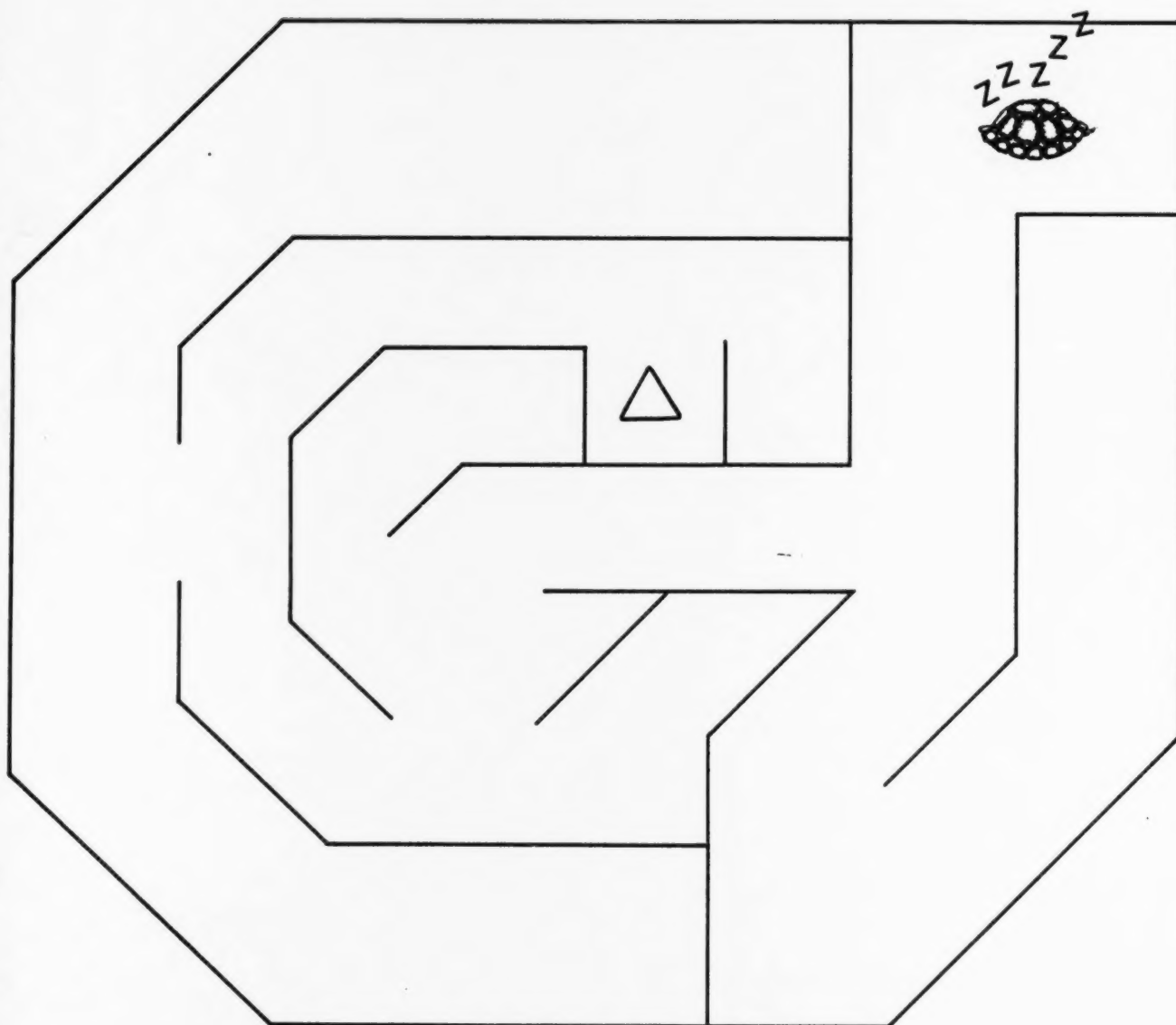
**N = NORTH
S = SOUTH
E = EAST
W = WEST**

**N.E = NORTHEAST
S.E = SOUTHEAST
S.W = SOUTHWEST
N.W = NORTHWEST**

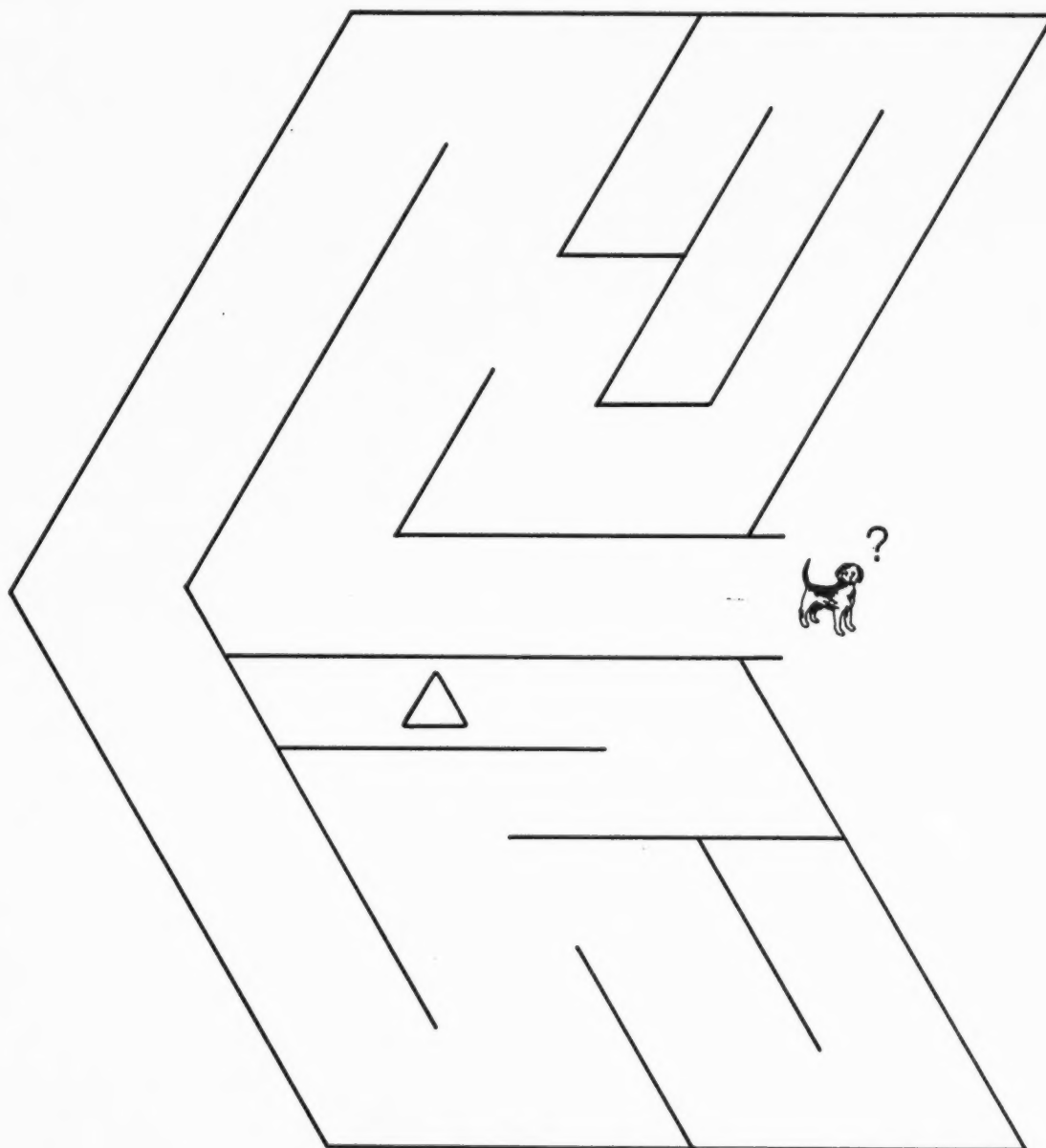
Map #1 for 'Using Compass Directions'



Map #2 for 'Using Compass Directions'



Map #3 for 'Using Compass Directions'



USING A DATA BASE

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "DATABASE

Be patient. It takes awhile for the procedures to load.

You are going to use the Logo procedures you have loaded to collect data. First, choose some data that you would like to collect. You might pick the name and phone number of each student in your class or important historical events and their dates.

Begin by typing

CREATE

You will see

FIELD . NAME ?

A field is one of the things that go in your data base. For example, if you are collecting names of your classmates and their phone numbers, then you might enter field names of STUDENT and PHONE. You can have as many fields as you want in your data base.

When you are done entering the names of the fields for your data base, you are ready to enter data into each record of your data base. A record is a collection of data, one entry for each field. Thus, if you are using your classmates' names and phone numbers, a record might be

MARY
555-2347

To enter records, type

ENTER . RECORDS

When you are finished, there are two ways that you can see what is in your data base. You can type

DISPLAY

and one record at a time will be displayed. If your data base is small, you can enter

DISPLAY . ALL

to see all of your records at once.

Is your data base complete and correct? If not, you can use

ADD

to add records at the end of your data base. Or, you can use

REMOVE

to remove unwanted records from your data base.

Would you now like a printed copy of your data base? If so, you can type

```
.PRINTER 1  
DISPLAY .ALL  
.PRINTER 0
```

Perhaps you would like to see all of the data in one field of your data base. For example, you might want to see just the names of everyone in your class. To do this, type

ONE.FIELD

If you would like the information from one field of your data base on the printer, you can type

```
.PRINTER 1  
ONE.FIELD
```

and then answer the questions. Then, be sure to type

```
.PRINTER 0
```

to turn the printer off.

This program allows you to work with only one data base at a time. If you want to save your collected data, you can type

SAVE.DATA

You will be asked for a file name. Be sure you write down the name you decide to use so you will remember it later. Then, be patient... this takes awhile.

At one point you will see the message

```
0 PROCEDURES SAVED
```

This means that just your data has been saved.

Once the prompt (?) and cursor return, you can either continue to use the same data, or you can type

CREATE

again and make a new data base.

If you want to load a data base that you created at some other time and saved on your disk, you can type

```
ERNS  
LOAD "<name you saved>
```

If you have a field that contains numbers, you might want to do a statistical analysis. You can do this by leaving the DATABASE program and moving to the STATISTICS program. To do this, type

ANALYZE

You will be asked which field you want to analyze. Then be patient... it takes awhile for all the files to load. A summary of the statistics commands is given below. (For more details, see the Statistics Section.)

SEE.LIST is used to see an existing list.
TOTAL gives the total of an existing list.
AVERAGE gives the average of an existing list.
SORT puts an existing list in order.
GRAPH prints a frequency distribution of an existing list using *'s
MAKE.LIST is used to create a new list.
ADD is used to add an element to an existing list.
REMOVE is used to remove an element from an existing list.

If you want to work with the data base program again, you must type

ERALL
LOAD "DATABASE"

Summary of commands to be used in the data base program:

CREATE is used to erase any previous data base and set up field names for the data base.
ENTER.RECORDS is used to erase any previous records and then enter data into the records in the data base.
ADD allows addition of one record at a time to the end of the data base.
REMOVE allows removal of one record from the data base and renumbers all the records after it.
DISPLAY shows one record at a time.
DISPLAY.ALL shows all of the data in the data base.
ONE.FIELD displays the data in one field of the data base.
SAVE.DATA allows just the data to be saved on disk.
ANALYZE moves one (numeric) field of the data base to the Statistics program for analysis.

For Logo programmers

The data for these procedures is always "stored" in the global variable DATABASE. You might want to change the values in a list using the editor instead of using ADD or REMOVE. To do this, type

EDNS

and make your changes in the editor. Be careful. DATABASE is a list of lists. Each record in the data base is a list and each field is a list within a list. When you are done, type Control/C and then you can analyze your modified list.

There are many rather sophisticated programming techniques used in the procedures in this activity. It would be instructive for you to see how each procedure works. In particular, you can learn a lot about OUTPUT, recursion, and recursive operations by studying the procedures in the workspace.

Teacher Information Sheet for Using A Data Base

Objectives:

- (1) Students will learn some of the terms associated with data base management systems.
- (2) Students will learn to use a computer to collect and organize data.
- (3) Programming students will learn more sophisticated ways of using Logo.

Grade level:

Students of all ages can use this program. However primary aged children should work with an older student or teacher who can do the typing.

Discussion:

This activity can be used with students and teachers who don't know any Logo. There are no programming skills needed. Instead, students can focus on data collection that is appropriate for the topic you are studying.

If you have students interested in programming, a lot can be learned just by studying how the procedures in the workspace operate. If you encourage students to explore the programming, be sure you have a "Logo expert" around to help answer questions!

Possible problems:

If Logo is already in a computer being used for this activity, be sure to type

ERALL

before loading the file. Otherwise, the file may be too large to fit in memory.

If your students want data on the printer, they are likely to make the following mistakes:

- the printer is not turned on
- the printer has no paper
- omitting the period in front of PRINTER
- spacing .PRINTER 1 and .PRINTER 0 incorrectly

If your students have no experience with Logo, they may become impatient and try to type responses when Logo is working. Caution them to wait for the prompt (?) and cursor (white square) to return to the screen after typing a command.

If your students generate very long databases, it may take Logo quite awhile to do some of the operations. You may even get an "OUT OF SPACE" error, in which case some data will need to be removed from the data base if you wish to continue.

USING STATISTICS TO ANALYZE DATA

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "STATISTICS

Be patient. It takes awhile for the procedures to load.

You are going to use the Logo procedures you have loaded to analyze some data. First, choose some data that you would like to study. You might pick the age of everyone in your class, the population of the ten largest cities in the world, or the high temperature for each day of the month.

Begin by typing

MAKE.LIST

You will see

TYPE STOP TO STOP ADDITIONS
ELEMENT 1 =

Enter your numbers, pressing return after each one. When you are ready to stop, type **STOP** instead of a number.

You can now see your list by typing

SEE.LIST

Your numbered list will appear. Is it correct?

You can add and remove numbers from your list. To add a number, type

ADD

You will see

AFTER ELEMENT #

Type the number of the element in the list. That is, if you have 10 numbers on your list, and you want to add a number after the 5th number on your list, type 5. (If you want to put the number at the beginning of the list, type 0.) Then type the number you want added to the list when Logo prints

ADD WHAT NUMBER?

Then, you can type

SEE.LIST

to see that your element is added correctly.

If you want to remove an element from the list, type

REMOVE

You will see

ELEMENT #

type the number of the element in the list, not the element itself.
That is, if you want to remove the 5th element, type 5.

Type

SEE .LIST

to see that your element is removed.

Make sure your list is correct. If you would like your list on the printer, you can type

.PRINTER 1
SEE .LIST
.PRINTER 0

Now you are ready to analyze your data.

To find the total of all of your numbers, type

TOTAL

To find the average, type

AVERAGE

If you would like your numbers in order, type

SORT

Be patient. If your list is long, this may take awhile. When the ? returns, type

SEE .LIST

to see your sorted list.

Finally, you can see a graph of your data by typing

GRAPH

If you would like any of your analysis on the printer, you can use the same method that was described above for printing your data. For example, to get the average on the printer, type

.PRINTER 1
AVERAGE
.PRINTER 0

or to get the graph on the printer, type

```
.PRINTER 1  
GRAPH  
.PRINTER 0
```

Summary of Statistics Commands:

MAKE.LIST is used to create a new list.
SEE.LIST is used to see an existing list.
ADD is used to add an element to an existing list.
REMOVE is used to remove an element from an existing list.
TOTAL gives the total of an existing list.
AVERAGE gives the average of an existing list.
SORT puts an existing list in order.
GRAPH prints a frequency distribution of an existing list using *'s

For Logo programmers

The data for these procedures are always "stored" in the global variable LIST. You might want to create a list or to change the values in a list using the editor instead of MAKE.LIST, ADD or REMOVE. To do this, type

EDNS

and make your changes in the editor. When you are done, type Control/C and then you can analyze your modified list.

You might want to change the characters used in the graph. You can do this by editing the procedure FREQUENCY. You could also add statements to procedure GRAPH to allow the user of the program to enter the symbol to use in the graph. Write any changes that you make here:

There are many rather sophisticated programming techniques used in the procedures in this activity. It would probably be instructive for you to see how each command works. In particular, you can learn a lot about OUTPUT, recursion, and recursive operations by studying the procedures in the workspace.

Teacher Information Sheet for Using Statistics To Analyze Data

Objectives:

- (1) Students will learn about collecting data for statistical analysis.
- (2) Students will learn how to analyze data.
- (3) Programming students can explore more sophisticated Logo programming.

Grade level:

This program can be used with students of any age. However, if you use it with primary children, either an older student or a teacher should help with the typing.

Discussion:

This activity can be used with students and teachers who don't know any Logo. There are no programming skills needed. Instead, students can focus on data analysis that is appropriate for the topic you are studying.

If you have students interested in programming, a lot can be learned just by studying how the procedures in the workspace operate. If you encourage students to explore the programming, be sure you have a "Logo expert" around to help answer questions!

NOTE: The SORT procedures are taken from the Logo Tool Kit produced by Logo Computer Systems, Inc., which is available to owners of Apple Logo.

Possible problems:

If Logo is already in a computer being used for this activity, be sure to type

ERALL

before loading the file. Otherwise, the file may be too large to fit in memory.

If you students want data on the printer, they are likely to make the following mistakes:

- the printer is not turned on
- the printer has no paper
- omitting the period in front of PRINTER
- spacing .PRINTER 1 and .PRINTER 0 incorrectly

If your students have no experience with Logo, they may become impatient and try to type responses when Logo is working. Caution them to wait for the prompt (?) and cursor (white square) to return to the screen after typing a command.

If your students generate very long lists, it may take Logo quite awhile to do some of the operations. You may even get an "OUT OF SPACE" error, in which case some numbers will need to be removed from the list if you wish to continue.

LOGO SCIENCE PROJECTS

"Logo Science Projects" and the associated files are © 1986 Logo Computer Systems, Inc. These are for personal use only, and are not to be copied or distributed without the express written permission of Logo Computer Systems, Inc.

Overview

This workshop is in three parts. First we'll review the structure of Logo. Logo grammar is generally very uniform. This consistency helps make it understandable and allows you to struggle less with Logo and concentrate more on the subject matter you are exploring.

Following this review, we'll develop Logo activities based on two topics that are part of the science curriculum at various grade levels. The first area is genetics and evolution. We'll simulate the classic pea plant experiment of Gregor Mendel. Second, the Turtle will be used to simulate various moving objects and provide us with a vehicle for exploring the laws of motion.

Mendel's Pea Plants

As the story goes, Gregor Mendel discovered some laws of genetics by mating tall and short pea plants. Mendel found that by mating short plants with tall plants he produced a second generation of plants that were all tall. Mating these with each other produced a third generation that was 25% short and 75% tall. What was going on?

Genes come in pairs. Each plant has a pair of genes for size. They may be both tall, both short, or one of each. The offspring of each pair of plants has a height gene from each parent. Each parental gene has an equal chance of ending up in the offspring.

Mendel figured that his original plants were "pure" short or tall since he had bred them for many generations within their own populations, always producing plants that were uniformly tall or uniformly short. Both height genes of the short plants carried shortness. The tall plants contained only genes for tallness. The height gene pair of the tall plants could be represented as TT, that of the short plants as SS.

By mating tall with short, all the plants of the second generation had the combination TS since they got one gene from a tall plant and one from a short plant. The plants were all tall because tallness is the dominant trait. These hybrid plants appear tall, but nevertheless carry a recessive gene for shortness.

By mating these hybrid plants with each other, there are four possible combinations of genes in the next generation: TT, TS, ST or SS. The first three result in tallness and account for 75% of the population in that generation. The other 25% are short.

Simulate this experiment in Logo. Here are some questions to guide you:

- How can you represent the gene pair? (Remember that Logo objects are words and lists.)
- How can you simulate random selection of one or the other gene from the pair?
- Having selected a gene from each of two individuals, how can you produce and represent the offspring?
- How do you produce and represent a large population of offspring?
- Can you make a Logo tool that tallies up how many of each type of offspring you've generated?

Once you've simulated Mendel's experiment, here are some more ideas to try:

- Assume that shortness (or tallness) is a trait that makes an individual more likely to survive long enough to reproduce. Have a substantial number of the members of your population die off based on their heights. Then mate the survivors. What happens to the proportions of tall and short plants in the next and subsequent generations?
- After favoring tallness or shortness for several generations, turn the tables and favor the other attribute. What happens to subsequent generations?

STOP HERE!

**READ NO FURTHER
LOAD NO FILES FROM YOUR FILE DISK
TRY IT YOURSELF FIRST**

Now, having worked on the above problems to your satisfaction and/or dismay, look at the discussion below and the procedures in the file MENDEL. This is how I used Logo for this exploration.

The first question is how to represent each individual according to its genes for height. I chose to use a word made of two characters, one for each gene. Each character may be T or S. A pure tall plant is TT. A pure short plant is SS. TS and ST represent hybrid tall plants.

(I could have used lists rather than words, [T S] or [TALL SHORT], for example, but words can be the names of things in Logo and I found this useful, as we'll see later on.)

The procedure OFFSPRING takes two gene pairs as input and produces a child:

```
?PRINT OFFSPRING "TT "SS
TS
```

PICK randomly picks one letter of each pair and these two are mated by WORD.

GENERATION creates a list of children from a specified population.

```
?PRINT GENERATION 10 [TT TT] [SS SS SS]
[TS TS TS TS TS TS TS TS TS TS]
```

Pure tall males and pure short females produce a generation of 10 hybrid tall plants. (Do the sizes of the initial populations affect the outcome?)

We can name the resulting generation:

```
?NAME GENERATION 100 [TT TT TT] [SS SS] "GEN1
```

will give the name GEN1 to the list of 100 elements, [TS TS TS TS ...].

This allows us to use TALLY to count how many of each kind of offspring we have.

```
?TALLY :GEN1
?PONS
MAKE "GEN1 [TS TS TS TS ...]
MAKE "TS 100
```

(Here's where it is useful to represent the gene pairs with Logo words rather than lists. TS can have a number as a value while [T S] can't.)

Now we can use GENERATION with GEN1 to produce the next generation of plants:

```
?NAME GENERATION 100 :GEN1 :GEN1 "GEN2
?ERN [TS]
?TALLY :GEN2
?PONS
MAKE "GEN1 [TS TS TS TS ...]
MAKE "GEN2 [TS SS ST TT TS SS TT ...]
MAKE "TT 23
MAKE "TS 29
MAKE "ST 24
MAKE "SS 24
```


Notice that the population GEN1 is the source of both males and females; any individual in the list may act as either. This is not an unreasonable assumption, at least for pea plants.

The command ERN [TS] is needed so that the tally of GEN2 doesn't include the tally of GEN1.

The results of TALLY will vary, but about 25% of the second generation will have each gene combination.

SELECT may be used to reduce a population based on differential survival rates that are determined by individual characteristics. Let's say we want to make tallness a trait that favors survival. We use ASSIGN.FACTOR to assign a survival probability to each gene combination.

```
?ASSIGN.FACTOR 90 "TT
```

will give a TT plant a 90% chance of surviving the effects of SELECT. ASSIGN.FACTOR uses Logo property lists, assigning TT the property SURVIVAL with the value 90. Try assigning TS and ST survival factors of 90 and SS a factor of 10. Then try

```
?NAME SELECT :GEN2 "GEN2
```

This reduces the size of GEN2.

```
?PRINT COUNT :GEN2
```

will reveal the reduced size of the population. Now try:

```
?ERN [TT TS ST SS].  
?TALLY :GEN2  
?PONS  
MAKE "GEN1 [TS TS TS ...]  
MAKE "GEN2 [ST TS TT TT ST SS ...]  
MAKE "TT 22  
MAKE "TS 26  
MAKE "ST 21  
MAKE "SS 4
```

or some other result that indicates the demise of short plants.

SELECT uses SURVIVAL.FACTOR to retrieve the SURVIVAL property from each member of the population. That number is compared with RANDOM 100. If RANDOM 100 turns out to be less than the survival factor, the individual is spared. This is quite likely if the survival factor is 90, but if it is 10, the plant is probably doomed.

Here are two more problems to attempt:

In the north of England there was a species of moths that were a speckled tan color. During the Industrial Revolution factories churned out a large amount of soot that blackened the trees in the area. The moth population became mostly black rather than speckled. The explanation is that the speckled moths were well camouflaged against the pre-industrial trees. Birds had trouble seeing and eating them. But as the trees darkened, blackness favored survival. In more recent times, the air has been cleaned up and thus, the trees are once again a lighter brown color. The moth population has also lightened, returning to its earlier speckled tan look. Create a Logo simulation of this process.

How about this one?

"The winter of 1946-1947 in Europe was very cold. The ground was frozen solid for more than three months. The moles of Europe died at an especially high rate because they could not dig for their usual food in the frozen ground. Probably fewer than one out of every hundred moles survived the winter.

Herr Stein studied the moles and found that the smaller moles survived the winter best. He believed the small moles were able to survive more easily when there was not much food. Larger animals that needed more food could not find it. As a result, more of them died of starvation. In this case, natural selection favored small size. It was the smaller moles that had offspring, and in the next generation, most of the moles tended to be small like their parents."

- Excerpt from: Natural Selection,
Man: A Course of Study, (c) EDC 1970

Motion

Logo can provide a valuable addition to the science curriculum because the Logo Turtle is a good object to play with when exploring the laws of motion. We can change things quickly to test new ideas. Gravity-free and friction-free environments, which are hard to achieve using physical objects, may be simulated. The Logo procedures that move the Turtle around the screen are statements of the laws governing this motion.

First, let's look at how the Turtle's environment compares to the real world we are trying to simulate. The world is three dimensional. The Turtle's world is a flat, two-dimensional screen. (It is possible to create Turtle graphics in perspective and work in three dimensions, but we won't get into that here.) In many situations, at least one dimension can be ignored. For example, simulating a car driving on a straight, flat road may be treated as a one-dimensional problem.

Driving

Now, how do we set the Turtle in motion? We can move it with FORWARD, BACK, SETX, SETY and SETPOS. These commands each take a very short time to execute. Try

```
?REPEAT 999 [FORWARD 1]
```

The Turtle moves!

- Try changing the Turtle's speed.
- Use the WAIT command to slow things down, for example

```
?REPEAT 999 [FORWARD 1 WAIT 5]
```

- Try other ways of moving the Turtle. Write some procedures that may be used to create motion. Simulate a car driving on a road.
- Introduce friction into your simulation. Friction is a force that acts to slow down the motion of the object. A car will eventually come to a stop if you don't give it gas.

- Friction has more of effect at high speeds than at lower speeds. Can you work this into your simulation?
- Add a gas pedal to you car. Remember that you have to press the accelerator in order to speed up, but because of friction, you have to maintain pressure just to keep going at a steady speed.

After you've worked on these ideas for a while look at the procedures in the file MOVE.

SETUP draws a "road" and aims the Turtle to the right.

```
TO setup
cs pu
sety -70 seth 90
pd forward 280
pu sety -65
END
```

MOVE, MOVE2 and MOVE3 are three different ways of setting the Turtle in motion. They all rely on the fact that motion is the ratio of distance to time, as in "miles per hour" or "feet per second." In MOVE3 we adjust both DISTANCE and TIME. In the other procedures we rely on the computer's built-in timing and just manipulate DISTANCE. (Can you see any problems with this approach?)

```
TO move
forward 1
move
END
```

Try MOVE2 1, MOVE2 5, etc.

```
TO move2 :distance
forward :distance
move2 :distance
END
```

Try MOVE3 1 0, MOVE3 1 1, MOVE3 1 5, MOVE3 10 10, etc.

```
TO move3 :distance :time
forward :distance
wait :time
move3 :distance :time
END
```

MOVE4 adds friction to our model.

```
TO move4 :distance :friction
if :distance < 0 [stop]
forward :distance
move4 ( :distance - :friction ) :friction
END
```

MOVE4 is somewhat inaccurate. Friction has more of an effect at high speeds. MOVE5 fixes this.

```

TO move5 :distance :friction
if :distance < 0.001 [stop]
forward :distance
move5 ( :distance - :distance * :friction ) :friction
END

```

MOVE6 adds a gas pedal to the car. Press G to give it gas. Notice that at some point, pressing G at regular intervals maintains speed. (Add the command PRINT :DISTANCE as the first line of MOVE6 to see this.)

```

TO move6 :distance :friction
if :distance < 0.001 [stop]
if keyp [gas rc]
pr :distance forward :distance
move6 ( :distance - :distance * :friction ) :friction
END

TO gas :key
if ( uppercase :key ) = "G [make "distance :distance + 1]
END

```

Falling

- Now put the Turtle near the top of the screen and have it fall. Falling objects are under the influence of gravity. Gravity exerts a constant force in the downward direction. This steadily increases speed. (You might want to draw a floor for the Turtle to land on.)
- Change the falling procedure so that the Turtle bounces off the floor.

The file BOUNCE has procedures that are my simulations of falling and bouncing.

SETUP draws a floor for the Turtle to detect with DOTP.

```

TO setup
cs pu
sety -70
seth 90
pd
forward 280
pu sety -80 seth 90
pd forward 280
pu sety -75 pd fill
pu sety 110 seth 0
END

```

I also created a global variable called GRAVITY and gave it the value 0.1. This is an arbitrary Turtle unit. On earth, the force of gravity is 9.8 meters/second/second or 32 feet per/second/second.

```
MAKE "gravity 0.1
```

DROP makes the Turtle fall until it hits the ground.

```
TO drop :distance
  if dotp pos [print "splat!! stop]
  sety ycor - :distance
  drop :distance + :gravity
END
```

Try this:

```
?SETUP DROP 0
```

The input to DROP is 0 because it starts at rest.

BOUNCE causes the Turtle to rebound instead of stop when it hits the floor.

```
TO bounce :distance
  sety ycor - :distance
  if dotp pos [bounce - :distance stop]
  bounce :distance + :gravity
END
```

To see the effect of these changes, try, for example:

```
?SETUP BOUNCE 0
```

Driving Off a Cliff

- Now that you've made the Turtle fall and bounce, try moving it horizontally and vertically at the same time.
- Simulate throwing a ball in a straight horizontal line. Start by assuming no friction. The horizontal motion continues unabated while the vertical motion is affected by gravity.
- Simulate a car driving off a cliff.

The file CLIFF contains my solutions to these problems.

FALL and BOUNCE allow you to set the Turtle in motion horizontally.

```
TO fall :xdis :ydis
  if dotp pos [print "splat!! stop]
  setx xcor + :xdis
  sety ycor - :ydis
  fall :xdis (:ydis + :gravity)
END
```

```

TO bounce :xdis :ydis
setx xcor + :xdis
sety ycor - :ydis
if dotp pos [make "ydis (- :ydis)]
bounce :xdis (:ydis + :gravity)
END

```

If combined with a procedure like GROUND to provide the needed setup, you can experiment with the two inputs needed to see how they affect the Turtle's motion.

```

TO ground
cs pu sety -70 seth 90
pd forward 280
pu sety -80 seth 90
pd forward 280
pu sety -75 pd fill
pu setpos [0 40]
END

```

Driving off a cliff is simulated with SETUP, which draws an upper and lower level, and DRIVE, which sets the Turtle in motion horizontally and calls FALL when the edge of the cliff is reached. Try these procedures with different input values.

```

TO setup
ground
cliff
setpos [-130 97]
seth 0
END

```

```

TO cliff
pu setpos [-130 95]
seth 90 pd
repeat 2 [fd 100 rt 90 fd 10 rt 90]
pu setpos [-120 90] pd fill
pu
END

```

```

TO drive :xd :yd
if xcor > -80 [fall :xd :yd stop]
setx xcor + :xd
drive :xd :yd
END

```


FUZZY LOGO:
LOGO IN MATH

by
Brian Silverman and Michael Tempel

"Fuzzy Logo: Logo In Math" and the associated files are © 1985 Logo Computer Systems, Inc. These are for personal use only, and are not to be copied or distributed without the express written permission of Logo Computer Systems, Inc.

Logo provides idealized environments for exploring ideas. The Turtle Geometry Microworld is a streamlined setting in which people may become acquainted with geometry and mathematics more generally. Feedback is quick and accurate.

REPEAT 4 [FORWARD 100 RIGHT 90]

reliably and precisely produces a square. The idea of squareness, four equal sides and 90 degree corners, emerges from the process of telling the Turtle to draw a square.

But what if Turtle Geometry was not quite so clean and neat? What if FORWARD and RIGHT were somewhat inaccurate? What if these commands didn't work at all? Would Turtle Geometry lose its educational value, or would new learning possibilities emerge instead?

We began with two separate ideas. One was to create a Logo in which FORWARD, BACK, RIGHT and LEFT worked, but with a random error of up to ten percent. FORWARD 100 moves the Turtle forward anywhere from 90 to 110 steps. This is not unlike the real world in which reliable systems, including the computers on which Logo runs, are engineered using imprecise components. Our expectation was that people would develop strategies utilizing feedback to correct or manage the defects we introduced into Logo.

A second starting point was to make a Logo in which FORWARD, BACK, RIGHT and LEFT didn't work at all. By trying to rebuild Turtle Geometry people would become familiar with trigonometry and explore the relationship between the relative system of Turtle Geometry and the absolute Cartesian system, which is also part of Logo.

As we perpetrated our funny Logos on groups of teachers during workshops the two topics merged. We also got more than we expected. New ideas came up. The strategies that people used to cope with our broken Logos fell into four distinct categories:

1. Feedback - In trying to get to a particular point, make a move. Then see where you actually ended up as compared to where you wanted to be. Use this information to correct your position.
2. Law of Large Numbers - Make your move in many small steps rather than one big one. Lots of little errors tend to cancel each other out.
3. Start Again - Forget about the messed up Turtle Geometry commands and make your own accurate ones by using the Cartesian system and trigonometry. The Turtle Geometry primitives can be built out of SETPOS, SETHEADING, SIN and COS.
4. Undo the Damage - Figure out how we broke Logo and then fix it. This gets into some of the more obscure areas of Logo, such as burying procedures and redefining primitive procedures.

It may be argued that these topics should not all be together in one paper or in one workshop since they are not well related to each other. They are presented together here because they existed together in practice, in workshops.

Let's look at each of the four strategies more closely. We'll refer to our broken versions of the primitive Turtle commands as FUZZY.FORWARD, FUZZY.RIGHT, etc. to distinguish them from the well-behaved versions of these commands. In workshops we generally redefine the primitives themselves into fuzziness.

Feedback

Start with FUZZY.FORWARD 100. This leaves you somewhere near your goal. You can find out where you are by using the Logo reporter POS (or XCOR and YCOR). If the Turtle starts at HOME with a heading of 0, then

```
FUZZY.FORWARD 100  
PRINT YCOR
```

will move the Turtle forward about 100 steps. In a normal Logo FORWARD 100 from HOME would leave you at a YCOR of 100. In our Fuzzy Logo you're somewhere between 90 and 110.

Now we haven't tinkered with the reporters. POS, XCOR, YCOR, HEADING, etc. are all accurate. This is not unlike the real world. You can buy an instrument, at your local electronics supermarket, that measures resistance with far greater precision than the tolerances of its own resistors and other components.

Now that you know where you ended up you can plan your next move. Let's say that YCOR is 95. You're five steps from the goal, so just go FUZZY.FORWARD 5. But, you might argue, FUZZY.FORWARD is off by up to 10%, so most likely you'll miss again. This is true, but with a 10% error, FUZZY.FORWARD 5 will be off by half a step at most. Let's take a worst case; FUZZY.FORWARD 5 moves the Turtle 5.5 steps, high by the full 10%. Now you're only .5 steps away.

FUZZY.BACK .5 results in an actual step back of between .45 and .55 steps. This leaves you within .05 steps of the goal, an error of only 1/20th of 1%.

After one move the maximum error is 10%. The second move leaves you off by no more than 1%. After the third move you are within 0.1%. You can continue if you wish, but this accuracy is already greater than the resolution of the graphics screen.

Starting at HOME with a heading of 0 we can summarize this solution in a procedure:

```
TO FIXED.FORWARD :AMOUNT  
NAME :AMOUNT "GOAL  
REPEAT 3 [FUZZY.FORWARD :GOAL - YCOR ]  
END
```

Since YCOR starts at 0, the first FUZZY.FORWARD tries to go the full amount. In the second repetition the input to FUZZY.FORWARD is the difference between the goal and the Y coordinate that resulted from the first FUZZY.FORWARD. The third FUZZY.FORWARD uses the difference between the goal and the Y coordinate resulting from the second FUZZY.FORWARD.

Three repetitions are arbitrary. The more corrections, the greater the precision.

It is possible to overshoot the goal. In that case, YCOR will be greater than :GOAL. The input to FUZZY.FORWARD will be negative, so the Turtle will move back.

This version of FIXED.FORWARD only works when the heading is 0 and the starting Y coordinate is 0. Here's a more general solution:

```
TO FIXED.FORWARD :AMOUNT
NAME LIST (:AMOUNT * SIN HEADING) (:AMOUNT * COS HEADING) "GOAL
REPEAT 3 [FUZZY.FORWARD DISTANCE :GOAL]
END
```

Actually, the first line of this procedure is by itself a fix for FUZZY.FORWARD. Look at strategy number 3 below. Here's another approach using feedback:

```
TO FIXED.FORWARD :DIST
IF :DIST < 0 [RT 180 FIXED.FORWARD - :DIST RT 180 STOP]
NAME POS "START
IF :DIST < 0.01 [STOP]
FUZZY.FORWARD :DIST
FIXED.FORWARD :DIST - DISTANCE :START
END
```

FIXED.FORWARD needs a DISTANCE procedure:

```
TO DISTANCE :POINT
OP SQRT SUM SQ XCOR - FIRST :POINT SQ YCOR - LAST :POINT
END

TO SQ :N
OP :N * :N
END
```

This FIXED.FORWARD stops when we are within 0.01 Turtle steps of the goal. We could have insisted on greater accuracy by using 0.001. What is of more interest is that we have chosen to use an absolute value rather than a percentage of the original distance. We limit our absolute error to .01 Turtle steps. FIXED.FORWARD 100 would be precise to 0.01%. But FIXED.FORWARD 10 would only be accurate to within 0.1% and FIXED.FORWARD 1 within 1%. Here's a version that achieves an accuracy of 0.1% regardless of the distance being attempted:

```
TO FIXED.FORWARD :DIST
NAME 0.001 * :DISTANCE "TOLERANCE
FX.FD :DIST
END

TO FX.FD :DIS
IF :DIS < 0 [RT 180 FX.FD - :DIS RT 180 STOP]
NAME POS "START
IF :DIS < :TOLERANCE [STOP]
FUZZY.FORWARD :DIS
FX.FD :DIS - DISTANCE :START
END
```

This is not necessarily a better solution. You just have to decide whether absolute or percentage error is important to you. This depends upon what you're doing. If you plan to attend a concert that begins at 8 o'clock and you want to avoid waiting more than a few minutes at the theater, you will need to arrive within, let's say, five minutes either side of 7:55. If the concert begins in an hour, this represents a much higher percentage error than if the concert is a year from now, but the absolute error is what counts.

On the other hand, you may want to know how fast you can run over a given distance. If the distance were ten kilometers, an absolute error in measurement of ten meters would throw your calculation off by 0.1%. If the distance were 100 meters, the same ten meters would represent a 10% error. In this case, percentage error is more important.

Now there's still a minor bug in FIXED.FORWARD. You might not want to let the Turtle overshoot its goal because this leaves extra little lines sticking out of the corners of your drawings. There is a solution. Don't try to go FUZZY.FORWARD the full distance. Instead, try a distance that will bring you to your goal when a maximum error occurs on the high side. In other words, 110% of the amount you choose should bring you to your goal.

```
FUZZY.FORWARD 1 / 1.1 * :AMOUNT
```

will do the trick. Instead of an error of plus or minus up to 10%, you get results that range between about 82% of the goal and the goal itself. The trade off for not overshooting is that you potentially have a larger error to correct for. This really isn't a problem because you can achieve any degree of accuracy you need by making more moves toward your target.

This strategy also allows us to simplify FIXED.FORWARD. Since :DIST - DISTANCE :START will only be negative if FUZZY.FORWARD :DIST overshoot the the goal, we no longer need the first line of the procedure. Here's an appropriate modification of the version of FIXED.FORWARD that achieves an absolute level of accuracy.

```
TO FIXED.FORWARD :DIST
NAME POS "START
IF :DIST < 0.01 [STOP]
FUZZY.FORWARD 1 / 1.1 * :DIST
FIXED.FORWARD :DIST - DISTANCE :START
END
```

How can we fix FUZZY.RIGHT using feedback? Here's a solution that is similar to the fix for FUZZY.FORWARD:

```
TO FIXED.RIGHT :TURN
IF ABS :TURN < 0.01 [STOP]
NAME HEADING "START
FIZZY.RIGHT :TURN
FIXED.RIGHT :START - :TURN
END

TO ABS :NUM
OP IF :NUM < 0 [-:NUM] [:NUM]
END
```

There are many examples of feedback mechanisms in the real world. A thermostat controls room temperature in this way. The heating system just pumps out heat. It has no way of dispensing well-measured amounts. The thermostat can measure the actual temperature. When it get's above a certain

point, the heating system is shut down. This causes the temperature to drop, and when it reaches a predetermined low point, the heat goes back on. You can decide how precisely to control the heat by adjusting the interval between the "turn on" and "turn off" temperatures.

The amount of water going into a toilet's tank is controlled by feedback. After each flush, the tank must fill with a fixed amount of water, enough for the next flush, but not so much as to overflow. The amount of water is never actually measured. Instead, the tank just starts to fill. As the water level rises, the float attached to the shut off valve also rises, eventually shutting off the incoming water flow when it reaches a certain height.

Cruise controls on cars sense changes in speed and adjust the amount of gasoline being fed to the engine.

The average temperature of the world is a fairly pleasant 15 degrees Celsius. However, the Earth's surface heats unevenly, which results in wide variations from this mean over time and in different places. The system's attempts to even things out create weather.

Lieutenant Island in Welfleet Harbor on Cape Cod hosted populations of rabbits and foxes. The foxes controlled the size of the rabbit population by eating them. A well-fed fox population would increase in size and eat more rabbits. The reduction in the rabbit population would cause some foxes to go hungry and die before reproducing. The drop in the fox population would allow more rabbits to survive. This natural feedback mechanism kept the populations stable within fairly narrow limits until two years ago when an island resident shot all the foxes. The island is now overrun by rabbits and a new feedback mechanism may be coming into play, balancing the rabbit population with the availability of edible vegetation.

This raises an interesting question. When do stable feedback mechanisms break down? Often some additional factor is introduced into the system. If you open all your windows on a cold day, the temperature may drop far below the minimum setting of your thermostat.

Law of Large Numbers

When we first created our Fuzzy Logo we tried some familiar Turtle graphics activities.

```
REPEAT 4 [FUZZY.FORWARD 100 FUZZY.RIGHT 90]
```

produced something that looked like it was trying to be a square.

```
REPEAT 3 [FUZZY.FORWARD 100 FUZZY.RIGHT 120]
```

almost gave us a triangle. The we tried

```
REPEAT 360 [FUZZY.FORWARD 1 FUZZY.RIGHT 1]
```

which to our initial surprise, produced an almost perfect circle! Why?

Well, the 360 errors in FUZZY.FORWARD 1 and in FUZZY.RIGHT 1 tended to cancel each other. FUZZY.FORWARD 100 has an even chance of moving the Turtle anywhere between 90 and 110 steps. However, REPEAT 100 [FUZZY.FORWARD 1] is very unlikely to end up far off.

To help see why, let's flip some coins. One coin gives you two possibilities, Head or Tail. With two coins there are four possibilities (HH HT TH TT). With three coins there are eight combinations (HHH HHT HTH THH HTT THT TTH TTT). Extending the pattern we have:

coins	number of combinations with N heads										
	0	1	2	3	4	5	6	7	8	9	10
1	1	1									
2	1	2	1								
3	1	3	3	1							
4	1	4	6	4	1						
5	1	5	10	10	5	1					
6	1	6	15	20	15	6	1				
7	1	7	21	35	35	21	7	1			
8	1	8	28	56	70	56	28	7	1		
9	1	9	36	84	126	126	84	36	9	1	
10	1	10	45	120	210	252	210	120	45	10	1

Let's examine how this works with ten flips. First of all, how many possible outcomes are there for these ten flips? The first flip has two possible outcomes. For each of these two, there are two possible outcomes of the second flip, making a total of four possibilities (HH HT TH TT). With three flips there are eight combinations (HHH HHT HTH THH HTT THT TTH TTT). With ten flips there are $2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 = 1024$ possibilities.

Now what's the probability of getting about half heads as compared to getting mostly heads? Let's say that about half of ten coins is 4, 5 or 6, and that mostly heads is 8, 9 or 10. The probability of getting about half heads is $210 + 252 + 210$ out of 1024 or $672/1024 = 0.656$, a two-out-of-three chance. The probability of getting mostly heads is $1 + 10 + 45$ out of 1024, or $56/1024 = 0.055$, only a little better than one in 20.

Start Again

A third approach to working with our Fuzzy Logo is to forget about the broken Turtle commands and start again from scratch. Try to build new Turtle commands out of other parts of Logo.

Turtle geometry is a relative system. FORWARD moves the Turtle some amount from where it is. There is no reference to an absolute frame of reference. Similarly, RIGHT rotates the Turtle some amount from the direction in which it is headed. It doesn't matter which way it was pointing.

Another geometry that is also built into Logo is the Cartesian system. There is a frame of reference in which position is referred to by X and Y coordinates and orientation is indicated by headings of 0 to 360 degrees. SETPOS (or SETX and SETY) sends the Turtle to a given place on the screen, regardless of where it was. SETHEADING orients the Turtle independent of which way it was initially pointing.

There are also reporters that tell you where the Turtle is (POS XCOR and YCOR) and which way it is pointing (HEADING). The relative system of Turtle geometry can be built by using these reporters to tell you where you are and which way you're heading. Based on that information, the absolute commands may be used to change position and heading.

Let's start with the simple case in which the Turtle is at "home" (a position of [0 0]) and has a heading of 0.

We can write a `FIXED.FORWARD`:

```
TO FIXED.FORWARD :DISTANCE
  SETY :DISTANCE
END
```

This works only for the first move.

```
TO FIXED.FORWARD :DISTANCE
  SETY YCOR + :DISTANCE
END
```

works as long as the heading is 0. Another special case is when the Turtle has a heading of 90.

```
TO FIXED.FORWARD :DISTANCE
  SETX XCOR + :DISTANCE
END
```

In general, we need to distribute `:DISTANCE` proportionally into X and Y components depending upon heading. If heading is 0, then the full `:DISTANCE` is in the Y direction. At a heading of 5 degrees, there's some movement in the X direction, but it's mostly in the Y direction.

Trigonometry takes care of the apportioning of `:DISTANCE` into the proper X and Y components. In fact that's exactly what `SIN` and `COS` do for a living. A general `FIXED.FORWARD` is

```
TO FIXED.FORWARD :DISTANCE
  SETPOS LIST (XCOR + :DISTANCE * SIN HEADING)
             (YCOR + :DISTANCE * COS HEADING)
END
```

Undo the Damage

In presenting Fuzzy Logo, we generally give people a Logo disk with a "Startup" file which automatically creates the Fuzzy Turtle commands. When Logo is booted, it looks for a file named `STARTUP`. If such a file is found, it is automatically loaded. When any file is loaded, Logo looks for the special name `STARTUP`. If it is found and if it is the name of a list, that list is run as a list of instructions.

This means that you can automatically cause a file to load and procedures to run from booting Logo. We created a Startup file that contained the procedures needed to redefine `FORWARD`, `BACK`, `RIGHT` and `LEFT` into fuzziness and then erase most of the evidence of our crime. However, there were enough clues left for some people to figure out what we did.

Here's what was in the Apple Logo II version of our Startup file:

```
TO CFD :N
  OLDFD NEARLY :N
END
```

TO SETUP
MAKEOLDS
MAKENEWS
ER [SETUP MAKEOLDS MAKENEWS]
ER [CFD CRT CBK CLT]
ERN "STARTUP
END

TO MAKEOLDS
COPYDEF "FD "OLDFD
COPYDEF "BK "OLDBK
COPYDEF "RT "OLDRT
COPYDEF "LT "OLDLT
END

TO CBK :N
OLDBK NEARLY :N
END

TO MAKENEWS
COPYDEF "CFD "FD
COPYDEF "CFD "FORWARD
COPYDEF "CBK "BK
COPYDEF "CBK "BACK
COPYDEF "CLT "LT
COPYDEF "CLT "LEFT
COPYDEF "CRT "RT
COPYDEF "CRT "RIGHT
END

TO CLT :N
OLDLT NEARLY :N
END

TO CRT :N
OLDRT NEARLY :N
END

TO CLEFT :N
OLDLT NEARLY :N
END

TO CRIGHT :N
OLDRT NEARLY :N
END

TO CBACK :N
OLDBK NEARLY :N
END

TO CFORWARD :N
OLDFD NEARLY :N
END

```

TO NEARLY :N
OUTPUT .9 * :N + :N * 2.N3 * RANDOM 100
END

```

```

MAKE "STARTUP [SETUP BURYALL]

```

The last line of the file shows that **STARTUP** is the name of the list **[SETUP BURYALL]**, so these two procedures are run when the file is loaded.

SETUP first calls **MAKEOLDS**, which uses **COPYDEF** to make copies of the four Turtle commands. These copies have the prefix **OLD**. Then **MAKENEWS** copies our new procedures onto the original names. Each of these new procedures (**CFORWARD**, **CRIGHT**, etc.) uses an original Turtle command, now renamed with the **OLD** prefix, along with **NEARLY** to create a fuzzy version.

The next step is to erase the procedures **MAKEOLDS**, **MAKENEWS** and **SETUP**. Then **CFD**, **CBK**, **CRT** and **CLT** are erased. They're no longer needed because their definitions have been copied onto the original primitive names.

Then the name **STARTUP** is erased in the last line of **SETUP**. Even though it is erased, the list it names continues to run until completed. The primitive command **BURYALL** buries the procedures that have not been erased, **OLDFD**, **OLDBK**, **OLDRT**, **OLDLT** and **NEARLY**. This means that these procedures will not respond to the commands **EDIT**, **PO**, **POTS**, **POPS**, **POALL**, **SAVE** or **ERASE**. Although this hides our dirty work to some degree, knowledgeable Logo people have strategies for finding out what we did.

After **POALL** turns up a blank, some people try **UNBURYALL** and then look again. Even though part of the contents of this file has been erased, enough is left, especially **NEARLY**, to figure out what's going on.

POFILE "STARTUP will display the full contents of the file as it was created, just as listed above. (We could have prevented this by including the command **ERASEFILE "STARTUP** in the **STARTUP** list.)

People who are not familiar with **BURYing** or using **POFILE** may still stumble upon our procedures. This has often happened when someone stops a procedure during execution. Logo will tell you what procedure you are in when you press the **STOP** character. This may be **NEARLY** or **OLDFD**. With these names revealed, the procedures may be printed out and edited.

One may restore Logo to normalcy by using **COPYDEF** to restore the old definitions to their original names. For example

```

COPYDEF "OLDFD "FORWARD
fixes FORWARD.

```

Summary

We started with a broken Turtle, which did not seem to have much to do with anything outside of itself. As we played with it and gave it to others to play with, we found that we had created a microworld for exploring geometry, trigonometry, probability, statistics, feedback and things like that.

According to Bob Lawler, microworlds are built out of "neat phenomena" and "powerful ideas." We took a neat phenomenon as our starting point. It spun off a number of powerful ideas, some intended, others surprising.

LOGO IN
MATH AND SCIENCE

LOGO ARITHMETIC

Student Worksheet

Here are several arithmetic problems. In the first column, write your own answer. In the second column, write what Logo responds after you type in the original command.

Logo Command	Your Guess	Logo's Response
PR 12 + 15	_____	_____
PR 22 - 18	_____	_____
PR 18 - 22	_____	_____
PR 31 * 3	_____	_____
PR 42 / 7	_____	_____
PR REMAINDER 14 4	_____	_____
PR QUOTIENT 21 4	_____	_____
PR 14 = 89	_____	_____
PR 21 > 12	_____	_____
PR 13 < 333	_____	_____
PR PRODUCT 12 3	_____	_____
PR INT 5.772	_____	_____
PR ROUND 426.831	_____	_____
PR ROUND 13.22	_____	_____
PR SQRT 9	_____	_____

Teacher Information Sheet for Logo Arithmetic

Objectives:

- (1) Students will learn how to do arithmetic using Logo.
- (2) Students will practice a variety of arithmetic operations using Logo.

Grade level:

Although aimed at a middle school audience, this activity can be adapted to almost any grade level. Students do not need to know how to program.

Discussion:

Once students have learned how to write standard arithmetic using Logo, you can use standard worksheets and allow student to use Logo to check their work.

List of operations available in Apple Logo:

+ returns the sum

If you type **PR 3 + 4** Logo will print 7

SUM will also return the sum.

If you type **PR SUM 3 4** Logo will print 7

- returns the difference

If you type **PR 5 - 4** Logo will print 1

* returns the product.

If you type **PR 3 * 4** Logo will print 12

PRODUCT also returns the product.

If you type **PR PRODUCT 3 4** Logo will print 12

/ returns the quotient.

If you type **PR 10 / 2** Logo will print 5

If you type **PR 12 / 5** Logo will print 2.4

QUOTIENT will return only the whole number part of a division problem.

If you type **PR QUOTIENT 12 5** Logo will print 2
(Note that QUOTIENT will only work with whole number inputs.)

REMAINDER returns only the remainder of a division problem.

If you type **PR REMAINDER 13 5** Logo will print 3
(Note that REMAINDER will only work with whole number inputs.)

= checks to see if two numbers are equal.

If you type `PR 4 = 4` Logo will print `TRUE`

If you type `PR 5 = 4` Logo will print `FALSE`

< checks to see if the first number is less than the second.

If you type `PR 3 < 4` Logo will print `TRUE`

> checks to see if the first number greater than the second.

If you type `PR 7 > 10` Logo will print `FALSE`

INT returns the whole number part of a decimal number.

If you type `PR INT 3.56` Logo will print `3`

ROUND returns the whole number part of a decimal number rounded to the nearest whole number.

If you type `PR ROUND 3.56` Logo will print `4`

SORT returns the square root of a number.

If you type `PR SQRT 16` Logo will print `4`

Logo also has the trigonometric functions of `SIN`, `COS` and `ARCTAN`, which can be used in the same manner as the previous procedures.

If your students are familiar with negative numbers, you might want to look at how `INT` and `ROUND` work with negative numbers. Take time to let the students discuss if what Logo does is what they expected and if it's what they think Logo should do.

Some mathematicians might argue, for example, that `INT -4.3` should be `-5` because they are familiar with the "greatest integer function," which always gives the integer that is at or to the immediate left of the function input. While there are times this is useful, in most computer settings, the way Logo picks the nearest integer "in the direction of zero" is more useful.

`ROUND` is less likely to be controversial. While some students may feel that `-6.5` should round up to the right to give `-6`, most will likely agree that since `6.5` rounds to `7`, `-6.5` should round to `-7`. These questions provide fertile ground for introducing, however informally, the notion of absolute value. With it, the notion of larger, meaning "to the right of," and larger, in the sense of magnitude or absolute value, can both be seen to have value.

EXPLORING TRIANGLES

Student Worksheet

You will be exploring using the Turtle to draw triangles in this activity. To begin, be sure you have loaded LOGO into your computer.

Type

CS

Now, try to draw a square. What directions did you use to draw it?

Were they something like

```
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
```

Or were they perhaps something like

```
REPEAT 4 [FD 62 RT 90]
```

A triangle is a figure with three (3) sides.
Now try to draw a triangle.

Sketch your first try here.
Type CS to try again.

Sketch your second try here.
Type CS to try again.

You may need to try several times.

Did you finally get a triangle? _____

What did you discover about the angles (turns) that you had to use?

Fill in the blanks for a triangle with all three sides equal.

FD 50
RT _____
FD 50
RT _____
FD 50
RT _____

Can you draw this triangle using REPEAT? What did you use?

Now try to draw a triangle where the sides are not equal.

Type CS to try again. Write your directions below and make a sketch of your drawing.

Teacher Information Sheet for Exploring Triangles

Objectives:

- (1) Students will learn to explore and experiment using Logo.
- (2) Students will learn how to draw a triangle using Logo.

Grade level:

This activity is appropriate for almost any age level when students are beginning Logo. Younger children will probably need more guidance than is provided on the worksheet, while older ones may want to go on to make designs using triangles.

Discussion:

Logo lends itself well to this type of exploration. Students should be encouraged to invent hypotheses and then to try them. Once students have had some time to explore, a group discussion/instruction period should be used to be sure that all students have learned to draw a triangle. Perhaps not all your students will produce equilateral triangles; be prepared to discuss (and value) these differences.

You may get questions from older children who have been taught that an equilateral triangle has three 60 degree angles and are concerned that Logo seems to use three 120 degree angles. Explain that the Turtle turns like we do, through what in geometry are called external angles which are different from the 60 degree internal angles. It is important to resolve the difference between standard Euclidian geometry and Turtle geometry.

EXPLORING POLYGONS AND CIRCLES

Student Worksheet #1

To begin, load LOGO into your computer.
Put the files disk into the disk drive.
Now type

LOAD "POLYGONS

One of the most common polygons is the square. To create a square on the screen, type the following:

SHAPE 90 50

Did you get a square? _____

If not, check your typing and try again.

Do you know what the size of each angle is? _____

How long is each side? _____

Hopefully you said that each angle is 90 and each side is 50. These are the same numbers that you used when you typed SHAPE 90 50.

Using the above information, what does the first number represent?

What does the second number represent?

If you are not sure, type

SHAPE 90 120

Here is another, type

SHAPE 90 30

What would you type to get a square with sides 65 steps long.

SHAPE _____

Another common shape is the triangle. You can use the SHAPE procedure to draw a triangle. Try some different numbers to see if you can get a triangle.

Did you draw a triangle? _____

Do you need a hint? The angles needed for an equilateral triangle are 120. Can you use this with SHAPE to draw a triangle now? What did you type?

SHAPE _____

Next, try some other second numbers, such as:

SHAPE 120 50

SHAPE 120 75

SHAPE 120 120

After this exploration, you should know that the first number represents the number of degrees that the Turtle will turn, and the second number represents the distance of each side.

The following activities need to be completed using the SHAPE procedure to experiment with different polygons.

Fill in the following blanks to create the shapes requested.

Triangle (3 sides) **SHAPE** _____

Square (4 sides) **SHAPE** _____

Pentagon (5 sides) **SHAPE** _____

Hexagon (6 sides) **SHAPE** _____

Octagon (8 sides) **SHAPE** _____

Nonagon (9 sides) **SHAPE** _____

Decagon (10 sides) **SHAPE** _____

Did you discover anything about the figures?

Explore the figures using different sizes of sides.

Which numbers create a shape which looks like a circle?

SHAPE _____

What is the largest degree number (first number) and the smallest side number (second number) you can use to get a circle?

SHAPE _____

What is the smallest circle you can draw?

SHAPE _____

A thought question: At what point does a polygon become a circle? You may want to do some research on this question.

Student Worksheet #2

Load Logo into the computer.

Next put your files disk in the disk drive, and then type

LOAD "CIRCLES

To use this program, you will tell the Turtle how far apart (in degrees at the center point) you want to put dots.

To see this, type

LOCUS 20

The Turtle will place dots 20 degrees apart.

Type

LOCUS 10

How many dots are drawn? _____

Type

LOCUS 30

How many dots are drawn? _____

Type

LOCUS 45

How many dots are drawn? _____

Type

LOCUS 8

How many dots are drawn? _____

Type

LOCUS 6

How many dots are drawn? _____

Type

LOCUS 5

How many dots are drawn? _____

Type

LOCUS 3

How many dots are drawn? _____

What happens when you use numbers smaller than 3? _____

Why do you think this happens.

Is LOCUS 1 a real circle? _____

Is LOCUS 1 a better circle than LOCUS 2 or LOCUS 3?

Teacher Information Sheet for Exploring Polygons And Circles

Objectives:

- (1) Students will explore drawing polygons of different sizes and shapes.
- (2) Students will learn that what appears on the screen as a circle is really a many-sided polygon.
- (3) Students will learn about defining a circle by its distance from a center point.

Grade level:

Both activities are appropriate for middle school and junior high students, however the polygon activity should probably precede the circle activity.

Discussion:

Logo allows students to explore the concept of a "limit." In this case, the larger the number of sides, the closer a polygon gets to looking like a circle. You should plan some follow-up discussion so students clearly understand that what looks like a circle on the screen is really a many-sided polygon.

The circle activity allows student to then look at a different "definition" of a circle. Instead of considering a circle as a multi-sided polygon, they will see a circle as a collection of points. In this case, they can explore how close together the points need to be to appear to be a circle. The word Locus used to name the procedure here comes from classical mathematics which defines a circle as "the locus of all points located a given distance from a given point." Here the distance is what we usually call the radius of the circle and the "given point" is the circle's center.

Older students should be encouraged to think about what a circle "really" is. This can lead to discussions of infinity and circles being made of an infinite number of points.

Answer key to shape questions:

(Note that while the first number must match to have a correct answer, the second number may be whatever size the student wants.)

Triangle	(3 sides)	SHAPE 120	50
Square	(4 sides)	SHAPE 90	50
Pentagon	(5 sides)	SHAPE 72	50
Hexagon	(6 sides)	SHAPE 60	50
Octagon	(8 sides)	SHAPE 45	50
Nonagon	(9 sides)	SHAPE 40	40
Decagon	(10 sides)	SHAPE 36	40

COMPARING LENGTH

Student Worksheet

To begin, load LOGO into the computer.

Now put your files disk in the drive .

Type

LOAD "LENGTH

Now type

BEGIN

to start the program. Do you see the line on the screen?

Can you draw a line that is longer?

Try it now..... Type

F

to go forward and

S

to stop when your line is longer.

Did you get an answer saying YES, THAT IS LONGER ? _____

If not, type

A

to try again. If you make a mistake, type

BEGIN

to start again. See how close you can make the lines and still have your line longer. Try again if you like.

Now try to draw a line that is shorter than the one that is shown. Either type

A

to continue or type

BEGIN

to start again.

When you think you have a line that is shorter, type

S

Did you get the answer YOURS IS SHORTER ? _____

Try again if you like.

See how close you can get and still have a line that is shorter.

Now you are to draw a line that is the same length as the one given.
Be careful. It must be just right!

If you do it right, you will get the answer THE 2 LINES ARE THE SAME!!!

Now here is another way to make lines that are longer, shorter or the same. This time we'll use lines on the worksheet and lines on the screen. First type

CL
FD 30

and look at the line you drew on the screen. Now type

CL
FD 60

and you can see that you got a longer line this time. We'll use this way to work with longer, shorter and the same.

Line 1: Type CL. Then try different numbers after the FD. Try to make a line that is longer than this one:

(hold this paper up to the screen to compare your line with the one on the paper.)

You drew your line using FD _____

Line 2: Now type CL and draw a line that is shorter than

You drew your line using FD _____

Line 3: This time type CL and draw a line that is the same as

You drew your line using FD _____

Line 4: Type CL and draw a line that is the same as

You drew your line using FD _____

Line 5: Type CL and draw a line that is longer than

You drew your line using FD _____

Teacher Information Sheet for Comparing Length

Objectives:

- (1) Students will practice drawing lines and comparing their length with given lines.
- (2) Students will practice estimation skills as they draw lines to match those given.

Grade level:

This activity is geared towards the primary student.

Discussion:

This activity allows students to compare two lines. They can discover if their lines are longer, the same length or shorter than the ones on the screen or on the worksheet.

The CL procedure is needed to both clear the screen and to prepare the Turtle to draw horizontal lines from left to right. It is not a Logo 'primitive.' That is to say, it is not a command Logo will recognize outside of this workspace.

Other activities could include students working together, one at the computer, the other with pencil and paper. One draws a line and asks the other to draw a line the same, longer or shorter. Either one can go first.

When students are comfortable with horizontal lines, see if they can draw lines the same length when the lines are vertical or are on an angle. There are short Logo procedures in the workspace named CV and CA that are similar to CL, but they clear the screen and prepare the Turtle to draw vertical or angled lines.

EXPLORING PERIMETERS

Student Worksheet

To begin, load LOGO into your computer.
Now place your files disk in the drive.

Type

LOAD "PERIMETERS

Definition: The perimeter of an object is the distance around the outside of that object.

The perimeters of the objects you will be using are to be measured in Turtle steps. However, perimeters are often measured in centimeters, inches, meters, miles and so on.

Type

EX1

You should see a square on the screen. This square has four sides and four angles. Each side is 50 Turtle steps and each angle is 90 degrees.

Now try to draw a square that is the same as the one on the screen beginning with the Turtle commands

**FD 50
RT 90**

Keep typing FD and RT commands until you have made your square. If you must, you can type

EX1

to start over.

How many times did you move forward? _____

Each time you moved forward, how many spaces did you move? _____

How many total Turtle steps did you use to make the entire square? _____

Was your answer 200 Turtle steps? If not, try EX1 again.

You should have used the following commands

```
FD 50 -----  
RT 90  
FD 50  
RT 90  
FD 50  
RT 90  
FD 50 -----
```

----- You can see that $50 \times 4 = 200$
Therefore the perimeter of the square is 200.

Now, explore another common shape, the triangle.
Type

EX2

The Turtle is already facing the proper direction. Can you draw a triangle just like the one shown on the screen?

If you need to start over, just type

EX2

Here is a hint if you are having trouble. You must use a turn of 120.

What is the length of each side? _____

How many sides are there? _____

What is the perimeter of this triangle? _____

Did you say each side was 50 and the perimeter was 150? If so, you are ready to continue, if not, try EX2 again.

You are now ready to try EX3 Type

EX3

What is the name of this shape? _____

How many sides are there? _____

If each side is 30 steps, what is the perimeter? _____

Now try to draw this shape. Type

EX3

to start over. (Hint: You must turn 72 degrees.)

What if each side of this figure were 50 steps long?

How long would the perimeter be? _____

What if each side were 100 steps long?

Now how long would the perimeter be? _____

You are now ready to try the next activity. Type

EX4

This is a hexagon. How many sides are there? _____

How long is each side? _____

What is the perimeter of this figure? _____

Can you draw a hexagon just like the one shown?

(Hint: You must turn 60 degrees).

Can you draw a hexagon with each side 20 steps long? Type

EX4

to try this.

What is its perimeter? _____

Can you draw a hexagon with each side 35 steps long?

What is its perimeter? _____

Now you are ready to move on to the next activity. Type

EX5

What is the name of this shape? _____

If the length of the top is 50, then the length of the bottom is _____

If the length of the left side is 30, then the length of the right side is _____

The total length around this shape (perimeter) is _____

Now try to draw the same shape.

For the next activity, type

EX6

How many sides are there on this shape? _____

If each side is 30 steps, what is the perimeter of this object? _____

Now try to draw the same shape. Type

EX6

if you need to start over.

Teacher Information Sheet for Exploring Perimeters

Objectives:

- (1) Students will review the idea of perimeter.
- (2) Students will get concrete experience with perimeters by having the Turtle draw a number of shapes.

Grade level:

Elementary through junior high students can benefit by experimenting with perimeters. Younger students can do just the first few activities; older students, all of them.

Discussion:

Students often confuse the ideas of perimeter and area. This activity makes perimeter concrete, requiring students to "drive" the Turtle around the shape.

The approach used in these activities could be used with students creating their own geometric figures and finding the perimeters. By writing them as procedures, other students can try to find the perimeters. Challenge your more experienced students to write procedures for other students to use.

ESTIMATING DISTANCES AND ANGLES

Student Worksheet #1

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "EST.DIST

Next type

TRY .ONE

You see a point.
The Turtle is pointing at that point.
Guess how many steps the Turtle needs to take to reach the point.
Type your number and then press RETURN.

Try this game as many times as you like.

When you are ready to continue, type

CS
FD _____ (type some number of your choice after the FD)

to try to get the Turtle to the top of the screen.

Try it several times.

What number after FD came closest for you? _____

Now try to get the Turtle to the bottom of the screen.
Type

CS
BK _____ (put in the number you want to try as before)

What number came closest for you this time? _____

Type

CS
RT 90

Now use FD _____ again to move the Turtle to the right edge of the screen.

As you try it again, remember the RT 90 as well as CS each time.

What number worked best for you this time? _____

Type

CS
RT 90

If you use BK ____ this time, how many Turtle steps do you need to get to the left edge of the screen?

BK _____

What other special Turtle step values can you find?

Student Worksheet #2

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "EST.TURN

Then type

TRY.ONE

You see a dot on the screen.
The Turtle is pointed straight up.
Guess how far the Turtle must turn (to the right) to point at the dot.
(360 will turn the Turtle all the way around.)
Type the number you think is best and press RETURN.

The Turtle will turn and move in the direction you told it.

Did the Turtle come close to touching the spot? _____
Try this game as many times as you wish.

When you are ready to go on, answer the next few questions.

What happens to the Turtle if you type

CS
RT 90

What happens if you type

CS
RT 180

What happens if you type

CS
RT 270

What happens if you type

CS
RT 360

Can you make the Turtle turn to face a corner?
What number looks best to you for this

RT _____

What happens if you use a number greater than 360?

Teacher Information Sheet for Estimating Distances And Angles

Objectives:

- (1) Students will practice estimating distances on the computer screen.
- (2) Students will practice estimating angles of Turtle turn.

Grade level:

This activity is appropriate for all ages of elementary students. More difficult activities may need to be developed for older students who have had more experience with Logo.

Discussion:

Estimation is a difficult topic to teach, since it is too easy for students to measure before estimating. This activity requires that students use estimation skills. Encourage them to spend time with this activity so that they become increasingly proficient at estimating both distance and angle.

You might want to have students work in pairs on this activity. Students can make the activity a game where each student tries to explain how s/he came up with her/his estimate. Be sure to avoid excessive competition, and encourage students to formulate hypotheses and then test them.

COORDINATE GEOMETRY

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "AXES

You are going to have the Turtle plot points on the screen. First type

AXES

What happens?

Now type

DOT [50 50]

(Be sure to use the square brackets: [and])

Where is the dot?

This dot is in what is called Quadrant I.

Now type

DOT [-50 50]

Where is the dot?

This dot is in Quadrant II.

Now type

DOT [-50 -50]

Where is the dot?

This dot is in Quadrant III.

Now type

DOT [50 -50]

Where is the dot?

This dot is in Quadrant IV.

Look carefully at each of the number pairs in the list below. Guess which quadrant DOT will put each pair in. Then use DOT to see if you are right.
(Just write I, II, III, or IV in the blank spaces)

	Your guess:	Logo shows:
1. 20 -50	_____	_____
2. 10 20	_____	_____
3. -10 -30	_____	_____
4. 40 10	_____	_____
5. -40 50	_____	_____
6. -10 -30	_____	_____
7. 25 -35	_____	_____
8. -45 -30	_____	_____
9. 30 -40	_____	_____
10. 15 60	_____	_____

Check yourself: 1 is IV, 2 is I, 3 is III, 4 is I, 5 is II,
6 is -III, 7 is IV, 8 is III, 9 is IV, 10 is I

Can you give a rule to tell which quadrant a point is in?

Quadrant I _____

Quadrant II _____

Quadrant III _____

Quadrant IV _____

For exploration:

How big can the first value after DOT be? _____

How small can the first value after DOT be? _____

How big can the second value after DOT be? _____

How small can the second value after DOT be? _____

When you are done with this worksheet type

WRAP

so the Turtle can "wrap around" again.

For Logo programmers:

Write a procedure to create a "dot to dot" picture.

TO DOT.TO.DOT

END

What happens if you change each DOT command to SETPOS in your procedure?

Teacher Information Sheet for Coordinate Geometry

Objectives:

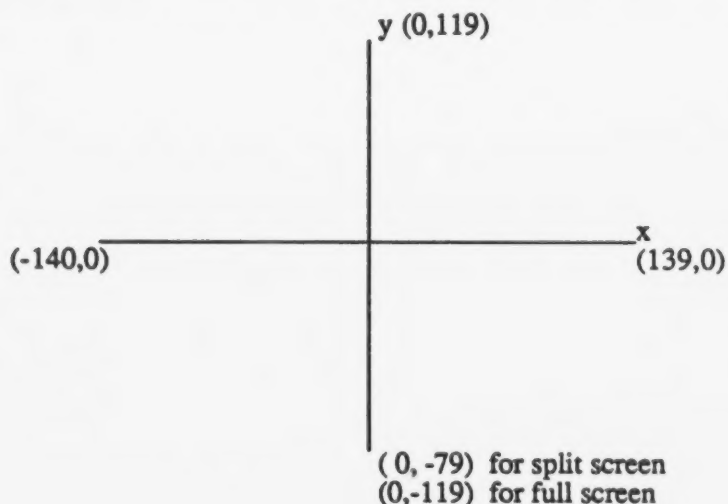
- (1) Students will learn the Logo commands DOT (and optionally SETPOS.)
- (2) Students will learn the numbering system for the four quadrants.
- (3) Students will practice plotting points.

Grade level:

This activity is appropriate when Cartesian geometry is introduced in math, probably at the fifth or sixth grade level. It could be used for review and practice for older students as well.

Discussion:

Logo has the capability of using Cartesian coordinate commands. The following is a list of the commands which can be used with the imaginary x and y axes that look like this. (The exact endpoints may vary among Logo dialects. This diagram applies to Apple Logo when FENCE is in effect.):



Summary of Commands and Operations:

- SETX number -- Moves the Turtle to the horizontal position represented by the number.
- SETY number -- Moves the Turtle to the vertical position represented by the number.
- SETPOS [number1 number2] -- Moves the Turtle to the point with x-coordinate 'number1' and y-coordinate 'number2' in a straight line.
- SETHEADING number -- Points the Turtle in the direction indicated by the number.
- DOT [number1 number2] -- Places a dot on the screen at the position having x-coordinate given by 'number1' and y-coordinate given by 'number2,' without moving the Turtle.
- PR TOWARDS [number1 number2] -- Accepts a list of two numbers as input and prints the direction that the Turtle must face to point toward the position represented by the two numbers.
- PR XCOR -- prints the current x-coordinate of the Turtle's position.
- PR YCOR -- prints the current y-coordinate of the Turtle's position.
- PR POS -- prints a list of the current x-coordinate and y-coordinate of the Turtle's position.
- PR HEADING -- prints the current direction in which the Turtle is facing.

EXPLORING SEQUENCES AND SERIES

Student Worksheet #1

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "SEQ.SERIES

Begin your exploration with this microworld by typing

ARITHMETIC

Perhaps you recall that an arithmetic sequence is one in which you get the next term by always adding the same amount to the latest term. This is the same as saying that the difference between each term and the next is always the same amount. So the program asks

FIRST TERM?

What number did you choose? _____

Next it asks for the

COMMON DIFFERENCE?

What did you choose for this? _____

Then it asks

HOW MANY TERMS?

you wish to have the program calculate and display.

This time you used _____

Then the program displays the number of terms you requested. Can you see the constant pattern of 'growth'? What is the value of the last term it displayed for you? _____
You can continue to experiment with this. Simply type

ARITHMETIC

each time you want to try another. You might like to see what happens if you use non-whole numbers, negative numbers, and more than just a few terms.

If you choose too large a number of terms, you can interrupt the program by pressing Control/G.

Tabulate some of your investigations here:

First	Difference	No. of terms	Last term
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

Student Worksheet #2

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "SEQ. SERIES

Geometric sequences are ones in which the next term is found by multiplying the latest term by the same number. Another way to say this is that the ratio of any term divided by the previous one is constant. In other words, they are like arithmetic sequences but use multiplication by a certain number instead of addition of a certain number.

Begin this investigation by typing

GEOMETRIC

When the program asks for your

FIRST TERM?

type a number of your choice. (What number did you choose? _____)

Then it asks for your choice of a

COMMON RATIO?

What number did you give it this time? _____

Finally it asks

HOW MANY TERMS?

Write the number you chose and the last term displayed on your screen.

Sometimes you will get very small or very large numbers. They are not written in the normal way. To see this, type

GEOMETRIC

and answer with a first term of 5, a ratio of 13, and ask for 12 terms. Look at the ninth term. It should be something like

4.07865E9

What that means is 4.07865 times 10 to the ninth power, or 4,078,650,000. Perhaps you would more quickly recognize it in the form:

4.07865 $\times 10^9$

In other words, when you see an 'E' near the end of a number it means take the number in front of the 'E' and multiply it by 10 as many times as the number after the 'E.' Now type

GEOMETRIC

again, and give a first term of 3, a ratio of .5, and ask for 10 terms. Here, the sixth term should look like

9.375N2 (meaning 9.375×10^{-2})

The 'N' here means use 10 to the negative power given after the 'N.' In this case, that means 9.375 times one one-hundredth, or .09375. If this brief explanation of Logo's use of the 'N' and 'E' to express small and large numbers doesn't make sense to you, ask for help before going on.

Now go ahead and experiment with GEOMETRIC yourself, writing some of your results in the following table.

First term	Ratio	No. of terms	Last term	Without N or E
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____

Student Worksheet #3

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "SEQ.SERIES

There are lots of other kinds of sequences besides arithmetic and geometric ones. One of the most common forms occurs when you can give a formula for the 'nth' term using 'n' and ordinary calculations. Try the following example.

Type

SEQUENCE

and then when asked:

DEFINE THE NTH TERM (IN TERMS OF N) :

type

N / (N + 3)

and then ask for 5 terms. Can you see that the numbers with which Logo responds are, in fact, its best representation of:

1 / (1 + 3)	-->	that is, 1/4
2 / (2 + 3)	-->	2/5
3 / (3 + 3)	-->	3/6 or 1/2,
4 / (4 + 3)	-->	4/7, and
5 / (5 + 3)	-->	5/8 ?

These sequences can be fun to play with, looking for interesting patterns. As before, record some of your investigations in the table below.

Definition of nth term	First 5 terms displayed				
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____

For the last part of this worksheet, look briefly at another kind of sequence. This kind is called a series instead of a sequence, and is formed from a 'regular' sequence by adding together the terms of the sequence. The elements or terms of the series are sometimes called 'partial sums' since they are sums of part of the original sequence. This will make more sense if you do an example.

Type

SEQUENCE

and then give the rule

$N * N$

and ask for six terms.

You should get the numbers 1, 4, 9, 16, 25, and 36.

Now type

SERIES

Can you guess what will happen if you again give the rule: $N * N$ and ask for six terms? Write the six numbers you actually got:

Can you see that the first is just the first term of the $N*N$ sequence, while the second is the sum of the first two $N*N$ terms ($1 + 4$), the third takes the first three terms ($1 + 4 + 9$), and so on?

Now it's time for you to experiment some more on your own. Can you find a series which doesn't just get bigger and bigger with each term, or one that doesn't just get bigger but also doesn't just get smaller? Summarize your experiments in the following table.

Definition of nth term	First 5 series terms displayed				
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____

Teacher Information Sheet for Exploring Sequences Snd Series

Objectives:

- (1) To give students a chance to experiment with several different kinds of sequences.
- (2) By removing the drudgery of hand (or even calculator) calculations, students are encouraged to explore these ideas more broadly and personally than is usually possible. The section on general sequences especially lends itself to recognizing individual discoveries of interesting sequences by naming them after the discoverer.
- (3) Each of the four topics presented can be extracted (and expanded) and used by itself to focus in more detail on that particular concept.
- (4) More complicated sequence rules can be examined than in more conventional approaches.
- (5) The idea of exponential or scientific notation can be introduced as a new topic, or its value illustrated by its necessity here.

Grade level:

The first two worksheets can be used at the junior high level when students may first see arithmetic or geometric sequences. The last worksheet is more appropriate for high school students.

Discussion:

- (1) Note that neither teacher nor students need to know anything about Logo in order to be able to use this microworld.
- (2) This can be used as a classroom demonstration tool.
- (3) With suitable computer availability, this can be the basis for student homework assignments.
- (4) Students can be challenged to develop a characterization of arithmetic sequence in the language of part three of the worksheet. (Geometric sequences can be pursued as well, but Logo does not have an exponential operation to permit direct implementation within this microworld)
- (5) Students can be challenged to think of ways to define sequences which while having clear patterns, do not fit within the schemes used in the worksheet (such as Fibonacci sequences).
- (6) Students who are familiar with Logo can be encouraged to modify and extend this exploratory environment in light of their understanding of and curiosity about the mathematics involved (including ideas 4 and 5 just above).

Possible problems:

- (1) If Logo has been in use just prior to this activity, it is advisable to use the command ERALL prior to the LOAD. Otherwise you may get an 'out of space' error message.
- (2) If students have not seen scientific notation before, some extra explanation may be needed. Also, students may encounter the need for this concept even in the arithmetic sequence section (before it is explained) if they choose large numbers.
- (3) If students get carried away, they may not only encounter scientific notation, but the inability of computer arithmetic to cope with all numbers. Logo will respond with a message NUMBER TOO BIG, or may give numbers as being zero that should really be very small but non-zero values.
- (4) Particularly in the sections with general sequences and series, students may give rules which will result in a division by zero. Logo responds with the message CANT DIVIDE BY ZERO and stops. Students may need help to see just what has happened, and how to circumvent the problem in a way most suitable to them.
- (5) An astute student may soon discover that Logo does not do all arithmetic exactly; it keeps only a few significant digits. For example, a geometric sequence starting at 1 with ratio .5 will begin to go slightly astray on the 10th term, giving 1.95312N3 instead of the exact value of 1.953125N3.
- (6) Students may enter syntactically incorrect sequence and/or series rules. Depending on the nature of their errors, various Logo error messages can result, or the function may simply be different than what was intended.
- (7) If students use Logo to expand this microworld, a Logo reference person may be needed for assistance.

EXPLORING LIMITS

Student Worksheet #1

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "LIMITS

By doing the activities suggested in this worksheet, you will be introduced to some ideas which may be new to you. These are but some of the intriguing possibilities mathematicians have in mind when they speak of "limits."

Begin by typing

FIGURE 3

What happened?

What do you think might happen if you type

FIGURE 4

After you have made a guess, try it to see.

What actually happened?

Now try

FIGURE 5
FIGURE 8
FIGURE 12

Try to describe what is happening as the number you type after the word **FIGURE** gets larger and larger.

What would you call the figure that is produced by **FIGURE 30**

Although to most people's eyes, FIGURE 30 looks like a circle, we can look at it in a different way to see that it is not actually a circle. FIGURE followed by the number N produces an N-sided polygon. The program is written so that the overall size of the figures remain about the same.

We can 'magnify' these shapes by using a different program that uses the same length for each side of each figure. For example, try

```
MAG.FIGURE 3  
MAG.FIGURE 5  
MAG.FIGURE 8  
MAG.FIGURE 12  
MAG.FIGURE 30
```

Can you see that even though the last figure goes off the screen, it is clearly made up of straight line segments, and so isn't a circle after all?

For most practical purposes with Logo, FIGURE 30 or FIGURE 60 would be perfectly acceptable to use as a circle, but mathematicians like to be sure that people realize that these are not really circles, but polygons. They have developed ways to describe the fact that the more sides there are, though, the "closer it is to being a circle." Limits express the sense of "closer and closer to, without ever quite getting there."

Student Worksheet #2

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "LIMITS

Imagine your friendly neighborhood Turtle standing with its nose 1 meter from the wall. Each time you tell it to "move," it is going to walk 1/2 of the distance which remains between it and the wall.

The program will report how far the Turtle is from the wall after having taken each walk you direct it to.

Begin by typing

GET . READY

What happens when you type

MOVE

Now type **MOVE** one or two more times. How far is the Turtle from the wall
now? _____

Can you rewrite this distance in centimeters instead of meters? _____

Do the move several more times. By now, the number which tells the Turtle's distance may look strange to you. You will see numbers like

6.25N2

meters. This is Logo's way of writing numbers which have several zeros after the decimal point. (It is a form of scientific notation for those of you who know about that.)

6.25N2 means 6.25×10^{-2}

That is, the N means that the following number is a Negative exponent of 10, and that the resulting power of 10 is to be multiplied by the number in front of the N. So 3.125N2 meters means .03125 meters (or 3.125 centimeters).

You can now type **MOVE** after **MOVE** after **MOVE** and watch as the Turtle gets closer and closer to the wall without ever quite reaching it.

If you want to start the Turtle over again, simply type **GET . READY** and then as many **MOVE**s as you want.

How close did your Turtle finally get to the wall? _____

The concept of limit applies here, again, since something is getting closer and closer to an 'obvious answer' but without ever actually getting there.

Now try to think of some similar phenomena; some situations where a process of a quantity keeps changing but by less and less, and where it seems to be getting closer and closer to some 'idealized' state or value. Write down your ideas.

Do you know how to program in Logo? If so, try to write your own program to demonstrate one of your ideas, or an idea of one of your classmates. Share your ideas and experiences with others.

Teacher Information Sheet for Exploring Limits

Objectives:

- (1) Students will experiment with two different processes which lead toward the concept of limits.
- (2) Being freed from the inaccuracy of pencil and paper, students will be able to pursue the limiting process further and gain more of an intuitive feeling for the infinite nature of limits.
- (3) Secondly, students will have an opportunity to review metric/metric conversions and scientific notation in the second part of the worksheet.

Grade level:

This activity is appropriate for junior high or high school students.

Discussion:

There are a number of ways that you might use this activity:

- (1) This can be used as a classroom demonstration project or as an individual or group project in a vein similar to a small science project.
- (2) More advanced students can be challenged to think of more examples and even to write demonstration programs for those situations. Some possibilities might include:
 - a. Calculate the areas of regular polygons inscribed in a unit circle as a method of approximating the area of a circle, or for approximating the value of π .
 - b. Using Turtle graphics, show a curve such as $y = x^2$ between $x=0$ and $x=3$ and a set of area-approximating rectangles below (and/or above) the curve, and then calculate such areas as the partition of the interval shrinks.
 - c. Examine the convergence of the partial sums of geometric series such as the sum of $1 + 1/2 + 1/4 + 1/8 + \dots + (1/2)^n$.
 - d. If you have access to IBM Logo, note that it permits you to set the precision of calculations. This can provide exciting capabilities to pursue limiting processes even further. Watch the reaction of calculus students when you set a program to compute successive Riemann sums for a simple definite integral. You may even want to let it run for a whole day or two!

Possible problems:

In the first part of the worksheet, covering polygons, students may naturally wonder what happens if they use fractional or negative values with FIGURE and MAG.FIGURE. Negative values will produce a Logo error message that looks something like:

REPEAT DOESN'T LIKE -5 AS INPUT IN ...

and no figure will result. Any positive number with a fractional part will produce no error messages, but the figure produced will not be a closed polygon.

In the second part, with the Turtle getting closer and closer to the wall, you will need to be prepared to explain scientific notation to those not already familiar with it.

A more serious problem will occur if students use more than 128 MOVE commands. At this point, the computer's arithmetic is unable to distinguish between the actual distance and zero, and actually prints that the distance is zero meters. The infinite nature of the limiting process, which is the central idea in this activity, will be less believable to students who see this result and believe that computers are infallible. Note, also, that it takes only a few steps before the number of significant digits in the distance exceeds the resolution of the computer's arithmetic, so those results are not fully accurate either.

One potentially interesting illustration of the inadequacy of computer arithmetic would be to write a program to generate a number in the trillions, and then add small numbers like 27 or 155 to it over and over, displaying the result each time. You should see that the values will stop changing. Clearly the real value continues to get larger, but the computer cannot follow it all the way because of the way it handles numbers of widely differing magnitudes.

Are some of your students attempting to write other limit demonstration programs? You may need to guide their ideas. Be sure that they are not planning projects that are too difficult for their abilities.

A CARTESIAN GRAPHING MICROWORLD

Student Worksheet #1

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "GRAPHING

Begin your exploration with this microworld by typing

SETUP

This will produce on your screen a set of Cartesian coordinate axes marked with tick marks intended to represent 1, 2, 3, etc. In both science and math it is common to have tables of values which one wants to display graphically in hopes of seeing some kind of pattern.

In this worksheet, you will use the computer to help you do this in several ways.

First, suppose that Ms. Smith's chemistry class of 23 students took a 10-point quiz with the following results:

<u>score on the quiz</u>	<u>number of students with that score</u>
2	1
4	4
5	6
6	5
7	4
8	1
10	2

To see this graphically, type the following lines.

PLOT 2 1
PLOT 4 4
PLOT 5 6
PLOT 6 5
PLOT 7 4
PLOT 8 1
PLOT 10 2

The next week she gave another 10 point quiz which had these results.

<u>score on the quiz</u>	<u>number of students with that score</u>
4	1
5	1
6	4
7	8
8	5
9	3
10	1

To see these results in a different way, type

```
USE BLUE X'S
PLOT 0 0
TRAIL
```

and then

```
PLOT 4 1
PLOT 5 1
PLOT 6 4
PLOT 7 8
PLOT 8 5
PLOT 9 3
PLOT 10 1
```

Can you see how the graphs show that more students found the first quiz harder than the second? Which way of displaying the data makes the data look clearer to you?

This brief worksheet has introduced you to a Logo microworld in which you can explore graphing points such as you might find when doing a science experiment or some mathematical operations. The commands that you have already seen are:

SETUP - which clears the screen and draws fresh axes.

PLOT a b - which plots the point (a,b) where 'a' and 'b' must be numbers.

BLUE, (and ORANGE and WHITE) - which determine the color of the points and any trail left between points.

X'S, (and SQUARES, CROSSES, and POINTS) - which determine the shape plotted at the point when PLOT is used.

TRAIL, (and NOTRAIL) - to turn on and off the 'leave a trail' feature.

USE - which actually does nothing but make some commands 'sound' more natural.

There is also a command TIDY.UP which you should use each time you are done with your use of this microworld, so the next person will find Logo as s/he normally expects to find it.

Now your teacher may have some additional sets of points for you to plot, or other activities using these graphing tools.

Student Worksheet #2

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "GRAPHING

Begin you exploration by typing

SETUP

and observe as a set of coordinate axes are drawn on the screen. In math and science classes you are often asked to draw graphs of equations or functions.

This microworld is intended to take some of the tedium out of the process of doing this kind of graphing.

For example, suppose you want to graph $y = 3 - x/2$. One way is to calculate several pairs of numbers which solve this equation and plot them, hoping to see a pattern. For now, just type

GRAPH

and then when Logo asks you to

GIVE THE FUNCTION RULE (IN TERMS OF X)

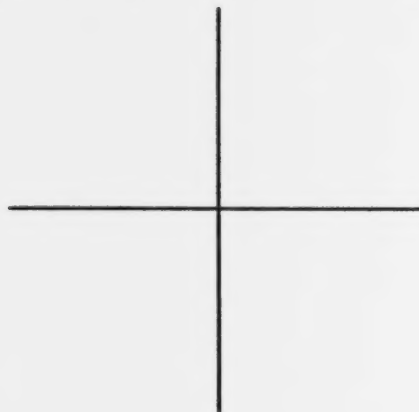
and prompts you with a

Y =

just type

$3 - (x / 2)$

and then watch as some points are plotted for you. Sketch what the graph looks like.



Next type

**USE WHITE
GRAPH**

and then when Logo responds with

GIVE THE FUNCTION RULE (IN TERMS OF X)

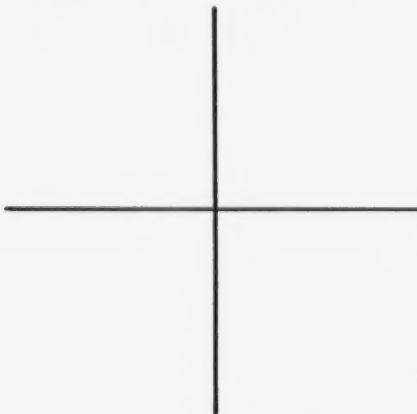
and asks

Y=

type

$(X * X / 5) - 6$

to see that Logo can cope with more complicated graphs, too. Sketch the result here:



You can also study families of functions. To see an example, type

SETUP

again to clear the screen and redraw the axes, then type

**TRAIL
GRAPH**

and then when Logo responds with

GIVE THE FUNCTION RULE (IN TERMS OF X)

and asks

Y=

type:

$0.5 * X$

When that graph is done, type

**USE BLUE
GRAPH**

and then give the simple rule:

X

Finally type

**USE WHITE
GRAPH**

and then give the rule:

2 * X

What happens to the graph in these cases as the coefficient of X gets larger? _____

This brief worksheet has introduced you to a Logo microworld in which you can explore graphing equations on the computer screen. The commands that you have already seen are:

SETUP - which clears the screen and draws fresh axes.

GRAPH - which prompts for the 'x portion' of an equation solved for y and then sketches a graph of the equation.

TRAIL, and NOTRAIL - which turn on and off the feature of having line segments drawn between the points which are calculated and plotted.

ORANGE, BLUE, and WHITE - which set the color of the points (and line segments) drawn.

USE - which does nothing more than make some of the commands more like spoken English.

When you are done using this microworld, please type TIDY.UP to restore Logo to its normal settings for the next person.

There are also a few other commands with which you may wish to experiment. Tell what each one does. (Hint: Not all have an immediate effect; you may have to graph something to see what is different.)

USE CROSSES _____

USE X'S _____

USE POINTS _____

USE SQUARES _____

QUICKLY _____

CAREFULLY _____

NORMAL SPEED _____

PLOT a b (where a and b are numbers, such as PLOT 7 -4)

Teacher Information Sheet for A Cartesian Graphing Microworld

Objectives:

- (1) To give students a chance to explore ideas using conventional Cartesian graphing concepts.
- (2) To encourage student thought about the curricular concepts beyond the mechanics required for graphing.
- (3) To provide a flexible tool suitable for a wide variety of specific graphing explorations.

Grade level:

The first worksheet can be used at the middle school level when students may first be expected to master Cartesian graphing skills. The second worksheet is more appropriate for high school students.

Discussion:

- (1) Note that neither teacher nor students need to know anything about Logo in order to be able to use this microworld. For those using the function graphing facility, it may be worth noting that:
 - (a) Multiplication must be denoted by an asterisk (*);
 - (b) In some dialects of Logo, the '/' for division needs a space on each side; and
 - (c) There are several built-in functions: at least SQRT(X), SIN(X), COS(X), ARCTAN(X), INT(X) are available in most Logo dialects. Note, though, that the trigonometric functions are based on degrees, not radians.
- (2) This can be used as a classroom demonstration tool.
- (3) With suitable computer availability, this can be the basis for student homework assignments.
- (4) Students can be challenged to come up with their own uses for this microworld, such as seeking trends in data that are of interest to them.
- (5) Students can be asked to formulate ways to rework data so that interesting characteristics can be more readily observed, such as combining several people's observations, averaging, or grouping. Various statistical and scaling techniques, both traditional and student invented, may be developed in this way.
- (6) Various characteristics of families of functions can be explored. The effects of various coefficients or combinations of coefficients can be the subject of experimentation prior to a formal, more rigorous development.
- (7) Students who are familiar with Logo can be encouraged to modify and extend this exploratory environment in light of their understanding of and curiosity about the mathematics involved.

Possible problems:

- (1) If Logo has been in use just prior to this activity, it is advisable to use the command ERALL prior to the LOAD. Otherwise you may get an 'out of space' error message.
- (2) Be prepared to handle the frustrations some students will experience with the limitations of graphical resolution. (Can you or a student perhaps modify the microworld to enable the user to permit 'magnification'?)
- (3) It is not difficult to give function rules for which some calculations may exceed the limits of computer arithmetic. In these cases Logo will issue the message NUMBER TOO BIG and stop. This can be a 'teachable moment' as much as it can be a stumbling block. If it is caused by dividing by a number close to zero, you could draw attention to the correlation of this fact with the impossibility of dividing by zero. If it is caused by a large exponentiation of some kind, you can look at just how quickly such expressions grow in comparison to low-order polynomial expressions.
- (4) In a vein similar to (3) just above, it is also easy to give rules which would require a division by zero. Logo responds with the message CANT DIVIDE BY ZERO and stops. Students may need help to see just what has happened, and how to circumvent the problem in a way most suitable to them. (You could also modify the microworld just a bit. As written it begins with $x = -13$. If you or a student altered it so that it started at, say, -13.01 , the x values used would less frequently equal the whole number values where division by zero is most likely to occur.)

- (5) Students may enter syntactically incorrect function rules. Depending on the nature of their errors, various Logo error messages can result, or the function may simply be different than what was intended.
- (6) If students use Logo to expand this microworld, a Logo reference person may be needed for assistance.

THE TURTLE IN MOTION

Student Worksheet

Load Logo into your computer.
Place your files disk into the disk drive.

Activity 1

Type

LOAD "MOVE1

You are going to experiment with a Turtle that moves by itself until you stop it. This is called a DYNAMIC TURTLE.

Type

MOVE 10 100

What happens?

Press Control/G to stop the Turtle. Then type

CS
MOVE 5 50

What happens? _____

Again you can press Control/G to stop the Turtle.

What does the first number stand for? _____

What does the second number stand for? _____

Experiment with other numbers. Can you get the Turtle to look like it does not stop between steps?

What were your directions to get this?

MOVE _____

Now you are ready to explore the next program.

Activity 2

Type

```
ERALL
LOAD "MOVE2
```

Type

```
MOVE 5 10 2
```

What happens? _____

When you are ready to stop, press Control/G. Now try

```
CS
MOVE 10 2 -1
```

What happens? _____

What do you think the third number represents? _____

If you said "acceleration," you used the correct technical word. The third number gives a force acting on the Turtle. It changes the Turtle's speed.

If you use a positive number for the acceleration, what happens to the Turtle?

If you use a negative number for the acceleration, what happens?

Experiment with this program using many different numbers. You may want to use 0 and decimals, also.

List any interesting results:

MOVE _____ The Turtle moved _____

MOVE _____ The Turtle moved _____

Try placing the Turtle at the top of the screen and making it fall to the bottom.

How did you do this?

Activity 3

Type

```
ERALL  
LOAD "MOVE3
```

Type

```
MOVE 2 4 20
```

What happens? _____

Press Control/G to stop, and then type

```
CS  
MOVE -4 -2 20
```

What happens? _____

Experiment with other values. Write your moves below and describe what happened.

MOVE _____ The Turtle moves _____

MOVE _____ The Turtle moves _____

MOVE _____ The Turtle moves _____

What have you discovered about the first number? _____

What have you discovered about the second number? _____

What does the third number represent? _____

This program allows you to move in something other than a vertical line.
What would you type to get the Turtle to move in a straight horizontal line?

MOVE _____

What would you type to get the Turtle to move in a straight vertical line?

MOVE _____

For a more sophisticated program, move on to the next part.

Activity 4

Type

```
ERALL  
LOAD "MOVE4
```

Type

```
MOVE 2 4 5
```

What happens? _____

Experiment with other numbers.

How does this program differ from MOVE3? _____

When you feel comfortable with this program, continue on to the next activity.

Activity 5

Type

```
ERALL  
LOAD "MOVE5
```

Type

```
MOVE 10 0 5 0 0
```

What happens? _____

Type

```
MOVE 10 0 5 .1 0
```

What happens? _____

Type

```
MOVE 0 0 5 .1 .2
```

What happens? _____

Experiment with other numbers. This program allows you to determine the acceleration (or deceleration) of both the x coordinate (horizontal value) and the y coordinate (vertical value).

What type of events could this program be used to simulate?

Write some interesting results you discovered:

MOVE	_____	_____	_____	The Turtle moved	_____
MOVE	_____	_____	_____	The Turtle moved	_____
MOVE	_____	_____	_____	The Turtle moved	_____
MOVE	_____	_____	_____	The Turtle moved	_____

Activity 6

Type

```
ERALL
LOAD "MOVE6
```

This program allows you to control the Turtle's motion from the keyboard while the MOVE procedure is running.

Type

```
F
```

to make the Turtle move faster. Type

```
S
```

to make the Turtle move slower.

Now try the program by typing

```
MOVE 5 10
```

and then experimenting with the speed.

What interesting results did you get while using this program?

Other Activities for Further Experimentation

So far the Turtle has moved, but only in straight lines. In STEER, DYNATURTLE and LUNAR.LANDER this is no longer so.

Steer

Begin by typing

```
ERALL
LOAD "STEER
STEER
```

and then follow the instructions on the screen. When you are done experimenting, press Control/G and then type

```
TIDY.UP
```

Dynaturtle

The ideas from STEER can be combined with some of those from the MOVE activities.

'DYNATURTLE' is one of the classic dynamic Turtle games.

R and L change the direction the Turtle is going, K provides a "kick" or acceleration.

To start the game type

```
ERALL
LOAD "DYNATURTLE
```

and follow the directions. You can stop and begin again by typing a period and then

```
STARTUP
DT
```

When you are all done experimenting, type a period and then

```
TIDY.UP
```

Lunar Lander

This is a rather challenging game in which you try to "land" the Turtle. To begin, type

```
ERALL
LOAD "LUNAR.LANDER
```

and follow the instructions on the screen. To restart the game you can type

```
INFO
```

(Remember:

R and L aim the ship, T provides the thrust.

At the bottom of the screen, you will see the x coordinate, y coordinate, horizontal and vertical speed.)

When you are done experimenting, type

```
TIDY.UP
```

Teacher Information Sheet for The Turtle In Motion

Objectives:

- (1) Students will learn about the dynamic Turtle.
- (2) Students will have an opportunity to interact with different kinds of Newtonian motion
- (3) Students will be able to experience and experiment with the effects of acceleration.

Grade level:

Experimenting with dynamic Turtles is applicable to students of any age level. However, the worksheets are geared toward students from middle school through high school. Younger students might use only the first two or three activities on the worksheet; older students may want to extend even the most complex of the activities.

Discussion:

One of the laws of science which is difficult to study is that of inertia. It states that a body in motion will continue in that state of motion (moving in a straight line or standing at rest) until some force acts upon it. To help children understand such concepts, as well as the relationship between distance and time and the effects of acceleration and friction, a dynamic Turtle can be designed and used.

The activities on the worksheet begin with the simplest form of motion and continue to involve various forces. Your students may not be ready for all the steps, but you may wish to explore them. While working with these procedures, keep in mind that the Turtle is acting as if it were floating in space or gliding on a smooth sheet of ice.

Once acceleration is introduced, some students may have difficulties grasping what is happening. Normally motion is affected by friction and students will expect that to be the case. This is probably a time to explain about the difference between Newtonian motion and the motion we experience.

BOUNCING TURTLES

Student Worksheet

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "BOUNCING

When the program is finished loading, type

DROP THE BALL

You will now be asked several questions about the conditions under which the ball will be dropped.

When Logo responds

WHAT ACCELERATION?

type 9.8

That is the amount of acceleration or "pull" of gravity.

When Logo responds

WHAT INITIAL HORIZONTAL VELOCITY?

type 0

This number is how "hard" you throw the ball "forward," to the right on the screen. The 0 means that you aren't throwing the ball at all. (Any number will work but with varying results.)

When Logo responds

WHAT IS THE COEFFICIENT OF ELASTICITY (0 TO 1)?

type .5

This is how "bouncy" the ball is. You can use any decimal number between 0 and 1. If you type 0, the ball will not bounce at all; if you type 1, it will never stop bouncing!

When Logo responds

WHAT INITIAL DOWNWARD VELOCITY?

type 0

This number tells how hard you throw the ball down. Using zero means that you just let go of the ball with no throw at all.

When Logo responds,

FROM HOW HIGH SHOULD IT BE DROPPED? (0 TO 236)?

type 100

This may be a positive number from 0 to 236 and indicates how close or far away from the bottom of the screen the ball will be dropped.

Now the Turtle will take over and put a dot at its position every "second" after being dropped. Pay careful attention to how the ball bounces.

When the ball stops bouncing, type

DROP THE BALL

again. This time use all the same responses as those described above except for the question

WHAT INITIAL HORIZONTAL VELOCITY?

This time type 2

Watch the effect of this change. What is the difference?

Experiment with this program, changing the values you input one at a time. How does each affect the way the ball bounces? Keep track of your inputs and note any interesting results.

Inputs					Results
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____

When you are done, please type

TIDY UP

to put Logo back in its normal condition for the next person.

Here are some other things that you can do with this program:

1. **DRAW THE DOTS BLUE**, followed by **DROP THE BALL**. You can also use **ORANGE**, **PURPLE**, **GREEN**, **WHITE**, and **BLACK**.
2. **DRAW THE LINE GREEN** and **DRAW THE PAGE WHITE** suggest other variations.
3. Use **CS** (or **CLEARSCREEN**) to clear the screen when it gets too cluttered.

Any combination of colors for the dots, background, and 'ground line' is possible. Also note that while the dotted path will not always reach all the way to the ground line, the dots from the bounce will nonetheless be in about the right place.

Teacher Information Sheet for Bouncing Turtles

Objectives:

- (1) Students will have an opportunity to experiment with a bouncing object.
- (2) Students will learn through experience the effects of gravity, horizontal and vertical velocity, and height on a bouncing ball.
- (3) Students will learn about the effect of the coefficient of elasticity on a bouncing ball.

Grade level:

Although fairly young children can experiment with this program, it is likely to be more useful with children from fifth or sixth grade through high school. To use it with young children, a simpler vocabulary would need to be used. Unless students wish to modify the program, no knowledge of programming is needed.

Discussion:

Logo's Turtle graphics can be put to use in a number of applications useful to the science teacher. The downward pull of gravity in our environment is something we grow up with but a careful analysis of the motion it produces is usually not studied outside the high school science classroom.

A falling object is used by most physics teachers for various studies. Without opening a window or going to the stairwell, you now have a falling object tool (the Logo Turtle) that can do many of the same things; except that this one bounces!

The BOUNCING program is recursive. A memory problem exists that will cause an OUT OF SPACE error to occur in tries that involve high coefficients of elasticity. Using the BOUNCING2 file instead of BOUNCING will help a little, but the color choices for dots, background, and line choices are gone, as are the "grace words" such as 'THE.' Some choices may still be made using Logo's commands of SETPC, SETBG, and PENUP or PENDOWN. One solution to the problem in Apple Logo is to run this program in Apple Logo II, which has more memory space.

You might be interested to know that some of the words used in the commands in this program are for us and not for Logo. THE, DRAW, and BALL, for example, sound right to us but are not needed. Also, the colors are words and not numbers; again, for the user's benefit.

High school science teachers should know that the program produces some variation from the "theoretical values." However, this variation is not unlike that produced when such an experiment is done in the lab.

CLASSIFICATION

Student Worksheet #1

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "PLANT.GAME"

The program automatically displays some introductory information. When you press Return, the program asks questions about plants that you must distinguish from one another.

Before you begin, choose a plant that you want to use for the first time you play the game.

Write the name of the plant here: _____

If it wasn't an plant that the program already knows about, what did you write to tell it from other plants?

Play the game for several other plants. Write each plant in the chart on the next page, and then complete the rest of the chart.

<u>Plant</u>	<u>Does the program know the plant?</u>	<u>Distinguishing Characteristic</u>
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Do you see that you need to plan just a bit in order to distinguish among the plants you've chosen?

List any problems identifying plants the program already knows:

List any problems you had describing plants you selected:

Student Worksheet #2

Load Logo into your computer.
Put the files disk into the disk drive.
Type

LOAD "ELEMENT . GAME

The program automatically displays some introductory information. When you press Return, the program asks questions about elements that you must distinguish from one another.

Before you begin, choose an element that you want to use for the first time you play the game.

Write the name of the element here: _____

If it wasn't an element that the program already knows about, what did you write to tell it from other elements?

Play the game for several other elements. Write each element in the chart on the next page and then complete the rest of the chart.

<u>Element</u>	<u>Does the program know the element?</u>	<u>Distinguishing Characteristic</u>
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Do you see that you need to plan just a bit in order to distinguish among the elements you've chosen?

List any problems identifying elements the program already knows:

List any problems you had describing elements you selected:

For Logo Programmers:

To make another game similar to the plant and element games, but for some other kind of object, be sure you have Logo loaded and then type:

```
ERALL  
LOAD "PLANT.GAME
```

or

```
ERALL  
LOAD "ELEMENT.GAME
```

if you prefer. Then type

```
UNBURY "PLANTS
```

(or UNBURY "ELEMENTS)

Next modify the procedure named INIT to start with a question appropriate to your list of items and give it one right answer and one wrong answer.

For example, for a tree game you might make INIT look like this:

```
TO INIT  
MAKE "TREE [[IS THIS TREE EVERGREEN?] SPRUCE MAPLE]  
END
```

There are other procedures in which you may wish to make changes. Once you are satisfied, remember to save your new game.

(Note: if your list needs a preceding A or AN, such as 'A MAPLE' the PLANT.GAME is the better starting place; if not, use the ELEMENT.GAME.)

Teacher Information Sheet for Classification

Objectives:

- (1) Students will gain practice in classification skills.
- (2) Programming students will gain some experience in working with lists.

Grade level:

The grade level varies with the way in which you use the program. While the plant game is suitable for most age levels, the element game is for students familiar with chemical elements. Students need little familiarity with Logo to use these programs unless they wish to modify them.

Discussion:

The Logo environment is especially suited for working with words and lists in a way that other computer languages cannot readily match. If you teach a subject in which lists of items must be classified and yet distinguished, this series of programs will make your students think about how and why objects are classified.

When using one of these games, you may want to give your students a list of objects and instruct them to play the game until Logo knows all the objects on the list. You could then compare different students' choices of distinguishing characteristics.

If you are using the Plant Game, you might want to start with:

Algae Moss Ivy Fern Pine.Tree Maple.Tree
Oak.Tree Rose Geranium Aloe Lichen

(Note: Spaces can't be used in names of plants.)

If you are using the Element Game, you might want to start with:

H He Ne Na Cl Al Fe Mg U Si O N

If a student has stopped the game but wants to resume with the same information, type

PLAY

(which will use the list of items left in the workspace) instead of

INFO

(which starts over with the original, built-in two items).

Students can save the information about their plants or elements by responding 'YES' when asked by the program

DO YOU WISH TO SAVE YOUR ... INFO?

They may then start a game with previously saved data by loading the game, interrupting it by pressing Control/G, then loading the file of previously saved data, and then typing **PLAY** instead of **INFO**.

Note: The files created by the plant game have the suffix ".PL.DA," and the files created by the element game have the suffix ".EL.DA."

Teachers or students who wish to develop further games like these two may find the following information helpful:

First, the procedures are "buried" so that when a player's list of objects is saved, only the list is saved, not the whole game. You need only unbury them as indicated on the student worksheet to be able to see what procedures are there. If you add more procedures, you'll need to look into property lists if you wish them buried along with the others in the final version of your game.

The ELEMENT.GAME is almost exactly like the PLANT.GAME, with the following changes:

- All occurrences (including inside other words) of SPECIES replaced by ATOM;
- All occurrences (including inside other words) of PLANT replaced by ELEMENT; and
- The procedure called A.OR.AN and any references to it were removed. In addition, the procedure INIT is as follows:

```
TO INIT
MAKE "TREE [X[IS THIS ELEMENT A NONMETAL?] OXYGEN IRON]
END
```

Note: These two games were adapted from an Animal Game found on the Logo Sample Files Disk, available from Logo Computer Systems, Inc.

APPENDIX 1

LISTINGS OF APPLE LOGO II FILES FOR TEACHER TRAINING ACTIVITIES

Indented lines occur in some of the following listings. These are for clarity only. Most versions of Logo require that you actually type the lines without pressing the return key.

PLURAL

```
TO LAST2 :WORD
OP LAST.CHARS 2 :WORD
END
```

```
TO LAST.CHARS :CHARS :WORD
IF :CHARS = 0 [OP "]
OP WORD ( LAST.CHARS ( :CHARS - 1 ) ( BL :WORD ) ) LAST :WORD
END
```

```
TO LAST3 :WORD
OP LAST.CHARS 3 :WORD
END
```

```
TO PLURAL :WRD
IF MEMBERP :WRD [MOUSE LOUSE] [OP WORD FIRST :WRD "ICE]
IF MEMBERP LAST2 :WRD [SH CH OX AX SS] [OP WORD :WRD "ES]
OP WORD :WRD "S
END
```

CONJUGATE

```
TO CONJUGER :PERSONNE :VERBE
IF MEMBERP :PERSONNE [JE IL ELLE] [OUTPUT SENTENCE :PERSONNE BUTLAST :VERBE]
IF EQUALP :PERSONNE "TU [OUTPUT SENTENCE :PERSONNE WORD BUTLAST :VERBE "S]
IF EQUALP :PERSONNE "NOUS
  [OUTPUT SENTENCE :PERSONNE WORD BUTLAST BUTLAST :VERBE "ONS]
IF EQUALP :PERSONNE "VOUS
  [OUTPUT SENTENCE :PERSONNE WORD BUTLAST BUTLAST :VERBE "EZ]
IF MEMBERP :PERSONNE [ILS ELLES]
  [OUTPUT SENTENCE :PERSONNE WORD BUTLAST :VERBE "NT]
END
```

```
TO CONJUGATE :PERSON :VERB
IF MEMBERP :PERSON [I YOU WE THEY] [OUTPUT SENTENCE :PERSON :VERB]
IF MEMBERP :PERSON [HE SHE IT] [OUTPUT SENTENCE :PERSON WORD :VERB "S]
END
```

SILLY

```
TO SILLY
OUTPUT (SENTENCE PICK :FAMILY PICK :VERBS PICK :NOUNS PICK :ADVERBS)
END
```

```
TO PICK :LIST
OUTPUT ITEM ( 1 + RANDOM COUNT :LIST ) :LIST
END
```

```
MAKE "ADVERBS [PASSIONATELY RESENTFULLY SLOPPILY]
MAKE "NOUNS [APPLES HATS COMPUTERS]
MAKE "VERBS [LOVED ATE POUNDED]
MAKE "FAMILY [MOMMY DADDY DAVID SUZY]
```

PHRASES

TO ARTICLE
OUTPUT RANPICK :ARTICLES
END

TO ADJECTIVE
OUTPUT RANPICK :ADJECTIVES
END

TO NOUN
OUTPUT RANPICK :NOUNS
END

TO RANPICK :LIST
OUTPUT ITEM (1 + RANDOM COUNT :LIST) :LIST
END

TO ADVERB
OUTPUT RANPICK :ADVERBS
END

TO MAKE.PHRASE :WORD1 :WORD2 :WORD3 :WORD4 :WORD5
PRINT :WORD1
WAIT 90 PRINT []
PRINT SE :WORD1 :WORD2
WAIT 90 PRINT []
PRINT (SE :WORD1 :WORD2 :WORD3)
WAIT 90 PRINT []
PRINT (SE :WORD1 :WORD2 :WORD3 :WORD4)
WAIT 90 PRINT []
PRINT (SE :WORD1 :WORD2 :WORD3 :WORD4 :WORD5)
END

TO VERB
OUTPUT RANPICK :VERBS
END

TO PHRASES
CLEARTEXT SETCURSOR [0 0]
MAKE.PHRASE ARTICLE ADJECTIVE NOUN VERB ADVERB
SETCURSOR [0 20]
PR [PRESS RETURN TO CONTINUE]
PR []
PR [PRESS PERIOD TO STOP]
IF RC = "." [STOP]
PHRASES
END

MAKE "ARTICLES [The Many Some]
MAKE "ADJECTIVES [cheerful bashful perky freckled tall]
MAKE "ADVERBS [quickly fleetingly recklessly happily glumly]
MAKE "NOUNS [birds children puppies gnus frogs]
MAKE "VERBS [tumble fly play gallop run]

POSTCARDS

```
TO POSTCARD
CLEARTEXT
REPEAT 3 [PR [ ]]
( PRINT [Dear] NAMES ", )
PR [ ]
PRINT BODY
PR [ ]
PRINT CLOSING
PR [ ]
PR NAMES
REPEAT 10 [PR [ ]]
WAIT 60
PR [Press RETURN to continue]
PR [ ]
PR [Press period to stop]
IF RC = ". [STOP]
POSTCARD
END
```

```
TO PICKRANDOM :LIST
OUTPUT ITEM ( 1 + RANDOM ( COUNT :LIST ) ) :LIST
END
```

```
TO BODY
OUTPUT PICKRANDOM :PHRASES
END
```

```
TO LEARN.NEW.PHRASE
PRINT [Please type in a new phrase]
MAKE "PHRASES FPUT READLIST :PHRASES
END
```

```
TO NAMES
OUTPUT PICKRANDOM :NAMES
END
```

```
TO LEARN.NEW.NAME
PRINT [Please type in a new name]
MAKE "NAMES FPUT READLIST :NAMES
END
```

```
TO CLOSING
OUTPUT PICKRANDOM :CLOSINGS
END
```

```
TO LEARN.NEW.CLOSING
PRINT [Please type in a new closing]
MAKE "CLOSINGS FPUT READLIST :CLOSINGS
END
```

```
MAKE "NAMES [E.T. Reta Mike Seymour Steve [Aunt Em] Occupant [Carl Krezdorn]]
MAKE "CLOSINGS [[Love] [See you soon.] [Don't forget to water the plants.]]
MAKE "PHRASES [[I love Logo.] [Hope everything's okay.] [Wish you were here.]
[Weather's great.] [Surf's up.] [Everyone's fine.]
[Don't forget to feed the cats.]]
```

POETRY

```
TO SENTEN :NOUNS :VERBS
IF OR EMPTY :NOUNS EMPTY :VERBS [PR [ ] STOP]
PRINT SE FIRST :NOUNS FIRST :VERBS
SENTEN BF :NOUNS BF :VERBS
END
```

```
TO ORDER.NOUNS
MAKE "NOUNS LPUT FIRST :NOUNS BF :NOUNS
OUTPUT :NOUNS
END
```

```
TO PICK.RHYME
OUTPUT ITEM ( 1 + RANDOM ( COUNT :RHYMES ) ) :RHYMES
END
```

```
TO ADD.NOUNS
PRINT [Type a list of nouns]
MAKE "NOUNS SE :NOUNS READLIST
END
```

```
TO ADD.RHYMES
PRINT [Type a list of rhyming words]
MAKE "RHYMES FPUT READLIST :RHYMES
END
```

```
TO POETRY
CLEARTEXT
SENTEN ORDER.NOUNS PICK.RHYME
IF RC = ". [STOP]
POETRY
END
```

```
MAKE "NOUNS [CHILDREN WORKERS FRIENDS MEN CARS BUSES BELLS BOOKS SPORTS WOMEN]
MAKE "RHYMES [[FLY TRY CRY BUY IMPLY [STAND BY]] [DEDUCT CONDUCT INSTRUCT OBSTRUCT]
[LEND MEND DEFEND PRETEND SPEND] [LAMENT REPENT REPRESENT INVENT DISSENT]
[OWN MOAN DRONE ALONE GROAN] [JOKE AWOKE SPOKE INVOKE PROVOKE]
[BITE FIGHT EXCITE REUNITE DELIGHT]]
```

TALK

```
TO TALK1
CLEARTEXT
PRINT [Hi. What's your name?]
PRINT [ ]
NAME READLIST "PERSON
PRINT [ ]
PRINT [Tell me what you did today.]
PRINT :PERSON
PRINT [ ]
NAME READLIST "ANSWER
PRINT [ ]
```

```

PRINT [That's interesting.]
PRINT [Tell me more.]
PRINT [ ]
NAME READLIST "ANSWER
PRINT [ ]
PRINT [It was nice talking to you. Bye.]
END

```

```

TO TALK2
CLEARTEXT
PRINT [Hi. What's your name?]
PRINT [ ]
NAME READLIST "PERSON
PRINT [ ]
PRINT SENTENCE [Tell me what you did today] :PERSON
PRINT [ ]
NAME READLIST "ANSWER
PRINT [ ]
PRINT [That's interesting.]
PRINT [Tell me more.]
PRINT [ ]
NAME READLIST "ANSWER
PRINT [ ]
PRINT SENTENCE [It was nice talking to you] :PERSON
PRINT [Bye now.]
END

```

```

TO TALK3
CLEARTEXT
PRINT [Hi. What's your name?]
PRINT [ ]
NAME READLIST "PERSON
PRINT [ ]
PRINT SENTENCE [Tell me] :PERSON
PRINT [What's your favorite season?]
PRINT [ ]
NAME READWORD "SEASON
PRINT [ ]
IF EQUALP UPPERCASE :SEASON "SUMMER [PRINT [I like hot weather too.] STOP]
IF EQUALP UPPERCASE :SEASON "WINTER [PRINT [Snow is great stuff.] STOP]
IF EQUALP UPPERCASE :SEASON "SPRING
    [PRINT [Yes it's nice to see the flowers grow.] STOP]
IF OR ( EQUALP UPPERCASE :SEASON "FALL ) ( EQUALP UPPERCASE :SEASON "AUTUMN )
    [PRINT SENTENCE [The trees are beautiful in] :SEASON STOP]
END

```

```

TO TALK4
CLEARTEXT
GREET
QUESTION
END

```

```

TO QUESTION
PRINT SENTENCE [Tell me] :PERSON
PRINT [What's your favorite season?]
PRINT [ ]
NAME READWORD "SEASON
PRINT [ ]

```

```

IF EQUALP UPPERCASE :SEASON "SUMMER [PRINT [I like hot weather too.] STOP]
IF EQUALP UPPERCASE :SEASON "WINTER [PRINT [Snow is great stuff.] STOP]
IF EQUALP UPPERCASE :SEASON "SPRING
    [PRINT [Yes it's nice to see the flowers grow.] STOP]
IF OR ( EQUALP UPPERCASE :SEASON "FALL ) ( EQUALP UPPERCASE :SEASON "AUTUMN )
    [PRINT SENTENCE [The trees are beautiful in] :SEASON STOP]
PRINT ( SENTENCE :SEASON [isn't a season] :PERSON )
PRINT [ ]
QUESTION
END

```

```

TO GREET
PRINT [Hi. What's your name?]
PRINT [ ]
NAME READLIST "PERSON
PRINT [ ]
END

```

```

TO TALK5
CLEARTEXT
GREET
QUESTION
KEEP.TALKING
SIGN.OFF
END

```

```

TO KEEP.TALKING
pr [ ]
( PRINT [Tell me about your favorite] :SEASON [sport.] )
pr [ ]
NAME READLIST "ANSWER
pr [ ]
END

```

```

TO SIGN.OFF
PRINT SENTENCE [It's been nice talking to you] :PERSON
PRINT [Bye for now]
END

```

LANGUAGE1

```

TO SQUARE
REPEAT 4 [FD 50 RT 90]
END

```

```

TO EVALUATE :THISLINE
IF :THISLINE = [BOX] [CS SQUARE STOP]
IF :THISLINE = [BALL] [CS CIRCLE STOP]
PRINT "HUH?
END

```

```

TO CIRCLE
REPEAT 36 [FD 5 RT 10]
END

```



```
TO SHAPES
TYPE "!"
EVALUATE READLIST
SHAPES
END
```

```
TO START
CLEARTEXT CS SHAPES
END
```

LANGUAGE2

```
TO SQUARE
REPEAT 4 [FD 50 RT 90]
END
```

```
TO CIRCLE
REPEAT 36 [FD 5 RT 10]
END
```

```
TO SHAPES
TYPE "!"
EVALUATE READLIST
SHAPES
END
```

```
TO START
CLEARTEXT CS SHAPES
END
```

```
TO EVALUATE :THISLINE
IF :THISLINE = [BOX] [CLEAN SQUARE STOP]
IF :THISLINE = [BALL] [CLEAN CIRCLE STOP]
IF :THISLINE = [UP] [PU SETY YCOR + 10 PD STOP]
IF :THISLINE = [DOWN] [PU SETY YCOR - 10 PD STOP]
IF :THISLINE = [RIGHT] [PU SETX XCOR + 10 PD STOP]
IF :THISLINE = [LEFT] [PU SETX XCOR - 10 PD STOP]
PRINT "HUH?"
END
```

LANGUAGE3

```
TO SHAPEP :WORD
OP MEMBERP :WORD [BOX SQUARE BALL CIRCLE TRIANGLE]
END
```

```
TO COLORP :WORD
OP MEMBERP :WORD [WHITE BLUE ORANGE RED GREEN VIOLET MAGENTA PURPLE]
END
```

```
TO TRIANGLE
REPEAT 3 [FD 60 RT 120]
END
```

```
TO BALL
CIRCLE
END
```

```
TO SQUARE
REPEAT 4 [FD 50 RT 90]
END
```

```
TO BOX
SQUARE
END
```

```
TO CIRCLE
REPEAT 36 [FD 4 RT 10]
END
```

```
TO EVALUATE :THISLINE
IF EMPTY? :THISLINE [STOP]
IF MEMBERP FIRST :THISLINE [CS CLEARSCREEN CLEAR]
    [CLEARSCREEN EVALUATE BF :THISLINE STOP]
IF MEMBERP FIRST :THISLINE [DRAW MAKE A AN THE] [EVALUATE BF :THISLINE STOP]
IF COLORP FIRST :THISLINE [SETPC THING FIRST :THISLINE EVALUATE BF :THISLINE STOP]
IF SHAPEP FIRST :THISLINE [RUN ( LIST FIRST :THISLINE ) EVALUATE BF :THISLINE STOP]
PRINT "HUH?
END
```

```
TO SHAPES
TYPE "!"
EVALUATE READLIST
SHAPES
END
```

```
TO START
CLEARTEXT CS SHAPES
END
```

```
MAKE "PURPLE 3
MAKE "MAGENTA 3
MAKE "VIOLET 3
MAKE "GREEN 2
MAKE "RED 4
MAKE "ORANGE 4
MAKE "BLUE 5
MAKE "WHITE 1
```

LANGUAGE4

```
TO AN
END
```

```
TO A
END
```

```
TO DRAW
END
```

```
TO PURPLE
SETPC 3
END
```

TO MAGENTA
SETPC 3
END

TO VIOLET
SETPC 3
END

TO GREEN
SETPC 2
END

TO RED
SETPC 4
END

TO ORANGE
SETPC 4
END

TO BLUE
SETPC 5
END

TO WHITE
SETPC 1
END

TO TRIANGLE
REPEAT 3 [FD 60 RT 120]
END

TO BALL
CIRCLE
END

TO SQUARE
REPEAT 4 [FD 50 RT 90]
END

TO BOX
SQUARE
END

TO CIRCLE
REPEAT 36 [FD 4 RT 10]
END

STORIES

TO STORY
textscreen ct
pr [Welcome to Logo!]
pr []
wait 60
pofile "chapter1
wait 180
cs square

```

wait 180
textscreen ct
pofile "chapter2
wait 180
cs triangle
wait 180
textscreen ct
pofile "chapter3
wait 180
cs house
wait 180
textscreen ct
pofile "chapter4
wait 240
village 12
ct
print [And they lived happily ever after]
END

```

```

TO STORY2
textscreen ct
pr [Welcome to Logo!]
pr [ ]
pr [( press any key each time you want to see more )]
ignore rc
ct
pofile "chapter1
ignore rc
cs square
ignore rc
textscreen ct
pofile "chapter2
ignore rc
cs triangle
ignore rc
textscreen ct
pofile "chapter3
ignore rc
cs house
ignore rc
cs textscreen ct
pofile "chapter4
print [How many houses do you want in the village?]
village readword
ct
print [And they lived happily ever after]
END

```

```

TO STORY3
textscreen ct
pr [Welcome to Logo!]
pr [ ]
pr [( press any key each time you want to see more )]
ignore rc
ct
pofile "chapter1
ignore rc
cs square

```

```

ignore rc
textscreen ct
pofile "chapter2
ignore rc
cs triangle
ignore rc
textscreen ct
pofile "chapter3
ignore rc
cs house
ignore rc
cs textscreen ct
pofile "chapter4a
branch rc
print [The end]
END

```

```

TO ignore :thing
END

```

```

TO spinhouse
repeat 36 [rt 10 house]
END

```

```

TO village :number
repeat :number [pu setpos list random 280 random 240 pd house]
END

```

```

TO house
square
fd 30 rt 30
triangle
lt 30 bk 30
END

```

```

TO triangle
repeat 3 [fd 30 rt 120]
END

```

```

TO branch :which
if equalp :which 1 [textscreen ct pofile "chapter5a ignore rc village 12 stop]
if equalp :which 2 [textscreen ct pofile "chapter5b ignore rc spinhouse stop]
pr [press the number 1 or the number 2]
branch rc
END

```

```

TO square
repeat 4 [fd 30 rt 90]
END

```

TEXT FILES USED BY 'STORIES':

CHAPTER1

Once upon a time,
there was a square.

CHAPTER2

... who met a triangle ...

CHAPTER3

They decided to get together
and make a house.

CHAPTER4

The house decided to get together
with some other houses to make
a village.

CHAPTER4A

What should the house do?

1. Get together with some other houses
to make a village?
2. Make a fancy design?

(press the number 1 or the number 2)

CHAPTER5A

The house decided to do something
about the housing shortage.

It got together with some other
houses to make a village.

CHAPTER5B

The house decided to spin around
and make a pretty design.

TEXT FILE USED WITH EDITOR

STORY

Once upon a time, in a far away and
distant land...

... and they lived happily ever after!

T H E E N D

TOOLS

```
TO big.replace :pairs :list
if empty? :pairs [op :list]
op replace2 ( first first :pairs ) ( last first :pairs ) big.replace bf :pairs :list
END
```

```
TO delete :word :list
if not memberp :word :list [op :list]
if equalp :word first :list [op bf :list]
op fput first :list delete :word bf :list
END
```

```
TO alter :file :wordout :wordin
open :file setread :file
change.lines :wordout :wordin
close :file
END
```

```
TO replace :wordout :wordin :list
if not memberp :wordout :list [op :list]
if equalp :wordout first :list [op fput :wordin bf :list]
op fput first :list replace :wordout :wordin bf :list
END
```

```
TO change.lines :wordout :wordin
if equalp readpos filelen :file [stop]
pr replace :wordout :wordin rl
change.lines :wordout :wordin
END
```

```
TO delete2 :word :list
if not memberp :word :list [op :list]
if equalp :word first :list [op delete2 :word bf :list]
op fput first :list delete2 :word bf :list
END
```

```

TO alter2 :file :pairs
open :file setread :file
change.lines2 :pairs
close :file
END

```

```

TO change.lines2 :pairs
if equalp readpos filelen :file [stop]
pr big.replace :pairs rl
change.lines2 :pairs
END

```

```

TO replace2 :wordout :wordin :list
if not memberp :wordout :list [op :list]
if equalp :wordout first :list [op fput :wordin replace wordout :wordin bf :list]
op fput first :list replace2 :wordout :wordin bf :list
END

```

```

TO alter3 :file :new.file :pairs
open :file setread :file
open :new.file setwrite :new.file
change.lines2 :pairs
close :file
close :new.file
END

```

TEXT FILE USED WITH TOOLS'

MOON

In the great green room
 There was a telephone
 And a red balloon
 And a picture of
 The cow jumping over the moon

MENDEL

```

TO pick :possibilities
op item ( 1 + random count :possibilities ) :possibilities
END

```

```

TO offspring :male :female
output word pick :male pick :female
END

```

```

TO generation :size :males :females
if :size = 0 [output [ ]] [output fput offspring pick :males
pick :females generation :size - 1 :males :females]
END

```

```

TO tally :stuff
if empty? :stuff [stop]
if not namep first :stuff [make first :stuff 1]
[make ( first :stuff ) ( thing ( first :stuff ) ) + 1]
tally bf :stuff
END

TO select :population
if empty? :population [output [ ]]
[if (random 100) < ( survival.factor first :population )
[output fput first :population select butfirst :population]
[output select butfirst :population]]
END

TO survival.factor :individual
output gprop :individual "survival
END

TO assign.factor :factor :indiv
pprop :indiv "survival :factor
END

```

MOVE

```

TO move
forward 1
move
END

TO setup
cs pu
sety -70 seth 90
pd forward 280
pu sety -65
END

TO move2 :distance
forward :distance
move2 :distance
END

TO move3 :distance :time
forward :distance
wait :time
move3 :distance :time
END

TO move4 :distance :friction
if :distance < 0 [stop]
forward :distance
move4 ( :distance - :friction ) :friction
END

```

```

TO move5 :distance :friction
if :distance < 0.001 [stop]
forward :distance
move5 ( :distance - :distance * :friction ) :friction
END

```

```

TO move6 :distance :friction
if :distance < 0.001 [stop]
if keyp [gas rc]
pr :distance
forward :distance
move6 ( :distance - :distance * :friction ) :friction
END

```

```

TO gas :key
if ( uppercase :key ) = "G [make "distance :distance + 1]
END

```

BOUNCE

```

TO setup
cs pu
sety -70 seth 90
pd forward 280
pu sety -80 seth 90
pd forward 280
pu sety -75 pd fill
pu sety 110 seth 0
END

```

```

TO bounce :distance
sety ycor - :distance
if dotp pos [bounce - :distance stop]
bounce :distance + :gravity
END

```

```

TO drop :distance
if dotp pos [print "splat!! stop]
sety ycor - :distance
drop :distance + :gravity
END

```

```

MAKE "gravity 0.1

```

CLIFF

```

TO fall :xdis :ydis
if dotp pos [print "splat!! stop]
setx xcor + :xdis
sety ycor - :ydis
fall :xdis ( :ydis + :gravity )
END

```

```

TO bounce :xdis :ydis
setx xcor + :xdis
sety ycor - :ydis
if dotp pos [bounce :xdis (- :ydis)]
bounce :xdis ( :ydis + :gravity )
END

```

```

TO setup
ground
cliff
setpos [-130 97] seth 0
END

```

```

TO cliff
pu setpos [-130 95]
seth 90 pd
repeat 2 [fd 100 rt 90 fd 10 rt 90]
pu setpos [-120 90] pd fill
pu
END

```

```

TO drive :xd :yd
if xcor > -30 [fall :xd :yd stop]
setx xcor + :xd
drive :xd :yd
END

```

```

TO ground
cs pu
sety -70 seth 90
pd forward 280
pu sety -80 seth 90
pd forward 280
pu sety -75 pd fill
pu setpos [0 40]
END

```

```

MAKE "gravity 0.1

```

FUZZY

```

TO FUZZY.LEFT :ANGLE
LEFT NEARLY :ANGLE
END

```

```

TO FUZZY.RIGHT :ANGLE
RIGHT NEARLY :ANGLE
END

```

```

TO FUZZY.BACK :DIST
BACK NEARLY :DIST
END

```

TO NEARLY :N
OUTPUT 0.9 * :N + :N * 0.002 * RANDOM 100
END

TO FUZZY.FORWARD :DIST
FORWARD NEARLY :DIST
END

APPENDIX 2

LISTING OF APPLE LOGO FILES FOR STUDENT ACTIVITIES

Indented lines occur in some of the following listings. These are for clarity only. Most versions of Logo require that you actually type the lines without pressing the return key.

FILES FOR LANGUAGE ARTS AND SOCIAL STUDIES

STORYTELLING

TO SPIDER
CS
PD
RSIDE
PU
HOME
PD
LSIDE
END

TO RSIDE
REPEAT 4 [RT 30 RLEG]
END

TO RLEG
FD 40
RT 90
FD 40
BK 40
LT 90
BK 40
END

TO LSIDE
REPEAT 4 [LT 30 LLEG]
END

TO LLEG
FD 40
LT 90
FD 40
BK 40
RT 90
BK 40
END

TO RAIN
PD
RT 123
SETBG 5
HT
REPEAT 600 [RAIN.DROP]
END

TO RAIN.DROP
FD RANDOM 20 PU FD RANDOM 20 PD
END

PLURALS

```
TO PLURAL :WORD.IN  
OUTPUT WORD :WORD.IN "S  
END
```

```
TO LAST2 :WORD.IN  
OUTPUT LAST.CHARS 2 :WORD.IN  
END
```

```
TO LAST3 :WORD.IN  
OUTPUT LAST.CHARS 3 :WORD.IN  
END
```

```
TO LAST.CHARS :CHARS :WORD.IN  
IF :CHARS = 0 [OUTPUT " ]  
OUTPUT WORD (LAST.CHARS (:CHARS - 1) (BUTLAST :WORD.IN)) LAST :WORD.IN  
END
```

SENTENCES

```
TO NOUN  
TYPE PICK :NOUNS  
SPACE  
END
```

```
TO VERB  
TYPE PICK :VERBS  
SPACE  
END
```

```
TO ARTICLE  
TYPE PICK :ARTICLES  
SPACE  
END
```

```
TO ADVERB  
TYPE PICK :ADVERBS  
SPACE  
END
```

```
TO SPACE  
TYPE [\ ]  
END
```

```
TO PICK :LIST.IN  
OUTPUT ITEM 1 + RANDOM COUNT :LIST.IN :LIST.IN  
END
```

```
TO .  
PRINT [.]  
END
```

MAKE "ARTICLES [THE A]
MAKE "VERBS [RUNS FLIES JUMPS WALKS EATS]
MAKE "NOUNS [CAT BIRD HORSE STUDENT]
MAKE "ADVERBS [QUICKLY EASILY GRACEFULLY]

SILLY.STORIES

TO STORY1
CLEARTEXT
PRINT [NAME A GIRL.]
NAME READLIST "GIRL
PRINT [NAME A PIECE OF FURNITURE.]
NAME READLIST "FURNITURE
PRINT [NAME A KIND OF FOOD.]
NAME READLIST "FOOD1
PRINT [NAME ANOTHER FOOD.]
NAME READLIST "FOOD2
PRINT [NAME AN ANIMAL.]
NAME READLIST "ANIMAL1
PRINT [NAME A VERB IN THE PAST TENSE.]
NAME READLIST "VERB1
CLEARTEXT
PRINT SENTENCE [LITTLE MISS] :GIRL
PRINT SENTENCE [SAT ON HER] :FURNITURE
PRINT (SENTENCE [EATING HER] :FOOD1 [AND] :FOOD2)
PRINT SENTENCE [ALONG CAME A] :ANIMAL1
PRINT (SENTENCE [AND] :VERB1 [BESIDE HER])
PRINT (SENTENCE [AND FRIGHTENED] :GIRL [AWAY.])
END

TO STORY2
CLEARTEXT
PRINT [NAME A NUMBER.]
NAME READLIST "NUM
PRINT [NAME THE PLURAL OF AN ANIMAL.]
PRINT [EXAMPLE: FOXES]
NAME READLIST "ANIMAL2
PRINT [NAME A VERB.]
NAME READLIST "VERB2
PRINT [NAME A FEMALE IN A FAMILY.]
PRINT [EXAMPLE: SISTER]
NAME READLIST "FEMALE
PRINT [NAME THE PLURAL FOR A PART OF THE BODY.]
PRINT [EXAMPLE: ARMS]
NAME READLIST "BODY
PRINT [NAME A SHARP OBJECT.]
NAME READLIST "SHARP
CLEARTEXT
PRINT (SENTENCE :NUM [BLIND] :ANIMAL2)
PRINT (SENTENCE :NUM [BLIND] :ANIMAL2)
PRINT SENTENCE [SEE HOW THEY] :VERB2
PRINT SENTENCE [SEE HOW THEY] :VERB2
PRINT (SENTENCE [THEY ALL] :VERB2 [AFTER THE FARMER'S] :FEMALE)

```
PRINT (SENTENCE [SHE CUT OFF THEIR] :BODY [WITH A] :SHARP)
PRINT [DID YOU EVER SEE SUCH A SIGHT IN YOUR]
PRINT [LIFE]
PRINT (SENTENCE [AS] :NUM [BLIND] :ANIMAL2)
END
```

NEWS.EXAMPLE

```
TO NEWS
MINUTE1
MINUTE2
END
```

```
TO MINUTE1
CLEARTEXT
PRINT [DALLAS COWBOYS WIN FIFTH GRADE MATH]
PRINT [FOOTBALL CONTEST]
PRINT [ ]
PRINT [ ]
PRINT [THE STUDENTS IN MRS. HALLENBECK'S]
PRINT [ ]
PRINT [CLASS WON THE FIFTH GRADE MATH]
PRINT [ ]
PRINT [FOOTBALL CONTEST ON FRIDAY, DECEMBER]
PRINT [ ]
PRINT [20, 1985 WHEN THEY BEAT MR. GOWER'S]
PRINT [ ]
PRINT [MIAMI DOLPHINS IN THE ALLPURPOSE ROOM]
PRINT [ ]
PRINT [AT MCDOWELL SCHOOL. THE SCORE WAS]
PRINT [ ]
PRINT [25 TO 14 WITH THE WINNING TOUCHDOWNS]
PRINT [ ]
PRINT [SCORED BY MATT FRAZER AND JEREMY]
PRINT [ ]
PRINT [STONE.]
NEXTSCREEN
END
```

```
TO MINUTE2
CLEARTEXT
PRINT [STUDENTS IN REACH DRAMA UNIT PERFORM]
PRINT ['MIDSUMMER NIGHT'S DREAM']
PRINT [ ]
PRINT [ ]
PRINT [FIFTH AND SIXTH GRADE STUDENTS IN]
PRINT [ ]
PRINT [THE REACH PROGRAM UNDER THE DIRECTION]
PRINT [ ]
PRINT [OF MRS. JOAN MAHER PRESENTED 'A]
PRINT [ ]
PRINT [MIDSUMMER NIGHT'S DREAM' FOR PARENTS]
PRINT [ ]
```

```

PRINT [AND FRIENDS IN THE HUDSON JUNIOR HIGH]
PRINT [ ]
PRINT [COMMONS ON MONDAY, JANUARY 13, 1986.]
PRINT [ ]
PRINT [THIS PLAY WAS THE FINAL PRODUCTION]
PRINT [ ]
PRINT [OF A SIX WEEK DRAMA UNIT.]
NEXTSCREEN
END

```

```

TO NEXTSCREEN
PRINT [ ]
PRINT [PRESS RETURN TO CONTINUE]
IGNORE RC
CLEARTEXT
END

```

```

TO IGNORE :THING
END

```

ADVERBS

```

TO ADVERB1 :WORD.IN
OUTPUT WORD :WORD.IN "LY
END

```

```

TO ADVERB2 :WORD.IN
IF LAST :WORD.IN = "Y [OUTPUT WORD BUTLAST :WORD.IN "ILY]
OUTPUT WORD :WORD.IN "LY
END

```

```

TO ADVERB3 :WORD.IN
IF AND ((LAST :WORD.IN) = "Y) (NOT MEMBERP LAST BUTLAST :WORD.IN [A E I O U])
[OUTPUT WORD BUTLAST :WORD.IN "ILY]
OUTPUT WORD :WORD.IN "LY
END

```

```

TO ADVERB4 :WORD.IN
IF AND ((LAST :WORD.IN) = "Y) (NOT MEMBERP LAST BUTLAST :WORD.IN [A E I O U])
[OUTPUTWORD BUTLAST :WORD.IN "ILY]
IF AND ((LAST :WORD.IN) = "E) ((LAST BUTLAST :WORD.IN) = "L)
[OUTPUT WORD BUTLAST :WORD.IN "Y]
OUTPUT WORD :WORD.IN "LY
END

```

```

TO ADVERB5 :WORD.IN
IF :WORD.IN = "GOOD [OUTPUT "WELL]
IF MEMBERP :WORD.IN [HARD FAST EARLY LAST] [OUTPUT :WORD.IN]
IF AND ((LAST :WORD.IN) = "Y) (NOT MEMBERP LAST BUTLAST :WORD.IN [A E I O U])
[OUTPUT WORD BUTLAST :WORD.IN "ILY]

```



```
IF AND ((LAST :WORD.IN) = "E) ((LAST BUTLAST :WORD.IN) = "L)
  [OUTPUT WORD BUTLAST :WORD.IN "Y]
OUTPUT WORD :WORD.IN "LY
END
```

LANGUAGE

```
TO DRAW
CS
END
```

```
TO A
END
```

```
TO SQUARE
REPEAT 4 [FD 60 RT 90]
END
```

```
TO TRIANGLE
REPEAT 3 [FD 60 RT 120]
END
```

```
TO RECTANGLE
REPEAT 2 [FD 60 / 2 RT 90 FD 90 RT 90]
END
```

COMPASS

```
TO N
PU
SETH 0
FD 10
END
```

```
TO W
PU
SETH 270
FD 10
END
```

```
TO E
PU
SETH 90
FD 10
END
```

```
TO S
PU
SETH 180
FD 10
END
```

TO S.E
PU
SETH 135
FD 10
END

TO N.W
PU
SETH 315
FD 10
END

TO S.W
PU
SETH 225
FD 10
END

TO N.E
PU
SETH 45
FD 10
END

DATABASE

TO CREATE
ERNS
MAKE "DATABASE []
MAKE "FIELD.NAMES GET.FIELD.NAMES []
MAKE "TOTAL.FIELDS COUNT :FIELD.NAMES
END

TO GET.FIELD.NAMES :NAMES
(TYPE [FIELD NAME?] CHAR 32)
MAKE "NAMES LPUT READWORD :NAMES
(TYPE [ANOTHER?] CHAR 32 (WORD "("Y "/" "N ")))
TEST READWORD = "Y
IFTRUE [OUTPUT GET.FIELD.NAMES :NAMES]
IFFALSE [OUTPUT :NAMES]
END

TO ADD
ENTER.LOOP
END

TO ENTER.LOOP
MAKE "DATABASE SENTENCE :DATABASE (LIST ENTER.A.RECORD 1 [])
(TYPE [ANOTHER?] CHAR 32 (WORD "("Y "/" "N ") CHAR 32)
IF READWORD = "Y [ENTER.LOOP]
END

```

TO TAKE.OUT :WHICH :CTR :LIST
IF EMPTY :LIST [OUTPUT [ ]]
TEST :WHICH = :CTR
IFTRUE [OUTPUT BUTFIRST :LIST]
IFFALSE [OUTPUT FPUT FIRST :LIST TAKE.OUT :WHICH :CTR + 1 BUTFIRST :LIST :LIST]
END

```

```

TO REMOVE
TYPE [RECORD #]
MAKE "DATABASE TAKE.OUT READWORD 1 :DATABASE
END

```

```

TO DISPLAY
CLEARTEXT
SETCURSOR [0 5]
DISPLAY.LOOP :DATABASE [ ] 1
END

```

```

TO DISPLAY.LOOP :LIST :FLAG :WHICH
IF EMPTY :LIST [STOP]
(PRINT [RECORD] (WORD "# :WHICH ":))
DISPLAY.REC FIRST :LIST 1
PRINT [ ]
IF NOT :FLAG = "ALL [CONTINUE]
DISPLAY.LOOP BUTFIRST :LIST :FLAG :WHICH + 1
END

```

```

TO DISPLAY.REC :ONE.REC :FIELD#
IF :ONE.REC = [ ] [STOP]
(PRINT (WORD ITEM :FIELD# :FIELD.NAMES ":) FIRST :ONE.REC)
DISPLAY.REC BF :ONE.REC :FIELD# + 1
END

```

```

TO DISPLAY.ALL
DISPLAY.LOOP :DATABASE "ALL 1
END

```

```

TO ENTER.RECORDS
MAKE "DATABASE [ ]
ENTER.LOOP
END

```

```

TO ENTER.A.RECORD :FIELD.NUMBER :RECORD
IF :FIELD.NUMBER > :TOTAL.FIELDS [OUTPUT :RECORD]
(TYPE ITEM :FIELD.NUMBER :FIELD.NAMES CHAR 32 [=] CHAR 32)
MAKE "RECORD LPUT READLIST :RECORD
OUTPUT ENTER.A.RECORD :FIELD.NUMBER + 1 :RECORD
END

```

```

TO ONE.FIELD
PRINT [WHICH FIELD NUMBER?]
SHOW.FIELDS 1
SHOW.DATA READWORD :DATABASE 1
END

```

```

TO SHOW.FIELDS :CTR
IF :CTR > (COUNT :FIELD.NAMES) [STOP]
(PRINT [FIELD] (WORD "# :CTR ":) ITEM :CTR :FIELD.NAMES)
SHOW.FIELDS :CTR + 1
END

```

```

TO SHOW.DATA :CTR :NAME :WHICH
IF EMPTY :NAME [STOP]
(TYPE [RECORD #] :WHICH ": [ ])
(PRINT [ ] ITEM :CTR FIRST :NAME)
SHOW.DATA :CTR BF :NAME :WHICH + 1
END

```

```

TO ANALYZE
(PRINT [WHICH FIELD NUMBER?] CHAR 32)
SHOW.FIELDS 1
MAKE "LIST GET.DATA READWORD :DATABASE [ ]
PPROP "LIST "VALPKG "LIST
PACKAGE "LIST "LIST
BURY "LIST
ERALL
UNBURY "LIST
LOAD "STATISTICS
END

```

```

TO GET.DATA :CTR :ALL :DATA
IF EMPTY :ALL [OUTPUT :DATA]
OUTPUT GET.DATA :CTR BF :ALL SENTENCE :DATA ITEM :CTR FIRST :ALL
END

```

```

TO SAVE.DATA
PPROP "RECORD "VALPKG "DATA
PPROP "DATABASE "VALPKG "DATA
PPROP "FIELD.NAMES "VALPKG "DATA
PPROP "TOTAL.FIELDS "VALPKG "DATA
PACKAGE "DATA "DATA
BURY "DATA
ERALL
UNBURY "DATA
REMPROP "RECORD "VALPKG
REMPROP "DATABASE "VALPKG
REMPROP "FIELD.NAMES "VALPKG
REMPROP "TOTAL.FIELDS "VALPKG
REMPROP "DATA "PROCPKG
(TYPE [UNDER WHAT NAME? \ ])
SAVE FIRST READLIST
LOAD "DATABASE
END

```

TO IGNORE :JUNK
END

TO CONTINUE
SETCURSOR [0 22]
(TYPE [PRESS RETURN TO CONTINUE...])
IGNORE READCHAR
CLEARTEXT SETCURSOR [0 5]
END

TO READWORD
OUTPUT FIRST READLIST
END

STATISTICS

TO MAKE.LIST
PRINT [TYPE \ \ STOP \ \ TO STOP ADDITIONS]
MAKE "LIST []
MAKE.LIST.LOOP 1
END

TO MAKE.LIST.LOOP :WHICH
LOCAL "NEW.ELEMENT
(TYPE [ELEMENT] CHAR 32 :WHICH CHAR 32 "=" CHAR 32)
MAKE "NEW.ELEMENT READWORD
TEST :NEW.ELEMENT = "STOP
IFFALSE [MAKE "LIST LPUT :NEW.ELEMENT :LIST MAKE.LIST.LOOP :WHICH + 1]
END

TO SEE.LIST
EXHIBIT :LIST 1 1 LAST CURSOR
END

TO EXHIBIT :LIST.NAME :COUNT :15LONG :ROW
TEST :COUNT > 15 * :15LONG
IFTRUE [MAKE "15LONG :15LONG + 1 MAKE "ROW (LAST CURSOR) - 15]
IF :COUNT > 15 [SETCURSOR LIST (:15LONG - 1) * 8 :ROW]
(PRINT WORD :COUNT ". FIRST :LIST.NAME)
TEST EMPTY BUTFIRST :LIST.NAME
IFTRUE [IF :15LONG > 1 [SETCURSOR LIST 0 (:15LONG * 15 - :COUNT + :ROW + 1)]]
IFFALSE [EXHIBIT BUTFIRST :LIST.NAME :COUNT + 1 :15LONG :ROW + 1]
END

TO REMOVE
TYPE [ELEMENT #]
MAKE "LIST TAKE.OUT READWORD :LIST 1
END

TO TAKE.OUT :WHICH :LIST :CTR
IF EMPTY :LIST [OUTPUT []]
TEST :WHICH = :CTR
IFTRUE [OUTPUT BUTFIRST :LIST]

```
IFFALSE [OUTPUT FPUT FIRST :LIST (TAKE.OUT :WHICH BUTFIRST :LIST :CTR + 1)]
END
```

```
TO ADD
LOCAL "AFTER
(TYPE [AFTER ELEMENT #])
MAKE "AFTER READWORD
(TYPE [ADD WHAT NUMBER?] CHAR 32)
INSERT :AFTER READWORD
END
```

```
TO INSERT :WHERE :WHAT
IF :WHERE = 0 [MAKE "LIST FPUT :WHAT :LIST STOP]
IF :WHERE = COUNT :LIST [MAKE "LIST LPUT :WHAT :LIST STOP]
TEST OR (:WHERE = 0) (:WHERE = COUNT :LIST)
IFFALSE [MAKE "LIST INSERT.LOOP 0 :WHAT :WHERE [ ] :LIST]
END
```

```
TO INSERT.LOOP :COUNT :TO.ADD :WHICH :PART1 :PART2
TEST :WHICH = :COUNT
IFTRUE [OUTPUT (SENTENCE :PART1 :TO.ADD :PART2)]
IFFALSE [OUTPUT INSERT.LOOP :COUNT + 1 :TO.ADD :WHICH (LPUT FIRST :PART2 :PART1)
(BUTFIRST :PART2)]
END
```

```
TO TOTAL
(PRINT [THE TOTAL IS] TOT :LIST)
END
```

```
TO TOT :LIST
IF (COUNT :LIST) = 1 [OUTPUT FIRST :LIST]
[OUTPUT ((FIRST :LIST) + (TOT BUTFIRST :LIST))]
END
```

```
TO AVERAGE
(PRINT [THE AVERAGE IS] ((TOT :LIST) / COUNT :LIST))
END
```

```
TO GRAPH
PRINT [.....WORKING.....]
MAKE "LIST SORT.IT :LIST
MAKE "TOGRAPH [ ]
MAKE "TEMP :LIST
GRAPH.LOOP
ERN "TEMP
CLEARTEXT
FREQUENCY
ERN "TOGRAPH
END
```

```
TO GRAPH.LOOP
IF (COUNT :TEMP) = 0 [STOP]
TEST (COUNT :TEMP) = 1
IFTRUE [MAKE "TOGRAPH LPUT LIST FIRST :TEMP 1 :TOGRAPH]
```



```

IFFALSE [MAKE.ELEMENT FIRST :TEMP 1 GRAPH.LOOP]
END

TO FREQUENCY
(TYPE FIRST FIRST :TOGRAPH CHAR 32)
REPEAT LAST FIRST :TOGRAPH [TYPE ""]
PRINT [ ]
MAKE "TOGRAPH BUTFIRST :TOGRAPH
TEST EMPTY :TOGRAPH
IFFALSE [FREQUENCY]
END

TO SORT
MAKE "LIST SORT.IT :LIST
END

TO SORT.IT :LIST.IN
IF EMPTY :LIST.IN [OUTPUT [ ]]
TEST EMPTY BUTFIRST :LIST.IN
IFTRUE [OUTPUT :LIST.IN]
IFFALSE [OUTPUT MERGE SORT.IT ODD :LIST.IN SORT.IT EVEN :LIST.IN [ ]]
END

TO EVEN :L
IF EMPTY :L [OUTPUT :L]
OUTPUT ODD BUTFIRST :L
END

TO ODD :L
IF EMPTY :L [OUTPUT :L]
OUTPUT FIRST :L EVEN BUTFIRST :L
END

TO MERGE :A :B :C
IF EMPTY :A [OUTPUT SENTENCE :C :B]
IF EMPTY :B [OUTPUT SENTENCE :C :A]
TEST (FIRST :A) < (FIRST :B)
IFTRUE [OUTPUT MERGE BUTFIRST :A :B SENTENCE :C FIRST :A]
IFFALSE [OUTPUT MERGE :A BUTFIRST :B SENTENCE :C FIRST :B]
END

TO MAKE.ELEMENT :THE.ELEMENT :HOW.MANY
TEST (COUNT :TEMP) = 1
IFTRUE [MAKE "TOGRAPH LPUT LIST :THE.ELEMENT :HOW.MANY
:TOGRAPH MAKE "TEMP BUTFIRST :TEMP STOP]
TEST :THE.ELEMENT = (ITEM 2 :TEMP)
IFTRUE [MAKE "TEMP BF :TEMP MAKE.ELEMENT :THE.ELEMENT (:HOW.MANY + 1)]
IFFALSE [MAKE "TOGRAPH LPUT LIST :THE.ELEMENT :HOW.MANY
:TOGRAPH MAKE "TEMP BUTFIRST :TEMP]
END

TO READWORD
OUTPUT FIRST READLIST
END

```

PROCEDURES FOR USING LOGO IN SCIENCE AND MATH

POLYGONS

```
TO SHAPE :A :B
CLEARSCREEN
WINDOW
PENUP
SETPOS [0 -75]
LEFT 90
PENDOWN
REPEAT (360 / :A) [FD :B RT :A]
WRAP
END
```

CIRCLES

```
TO LOCUS :TURN
CLEARSCREEN
PENUP
REPEAT (360 / :TURN) [PUT.A.DOT :TURN]
PENDOWN
END
```

```
TO PUT.A.DOT :TURN
RIGHT :TURN
FORWARD 79
PENDOWN
DOT POS
PENUP
BACK 79
END
```

LENGTH

```
TO BEGIN
DO.ONE.PROBLEM
PRINT [TYPE A IF YOU WANT TO TRY AGAIN]
IF READCHAR = "A [BEGIN]
END
```

```
TO DO.ONE.PROBLEM
CLEARSCREEN
MAKE "ORIG (5 * (1 + RANDOM 20))
GET.SET
MAKE "ST.ANS 0
STUDENT.DRAW
EVALUATE
END
```

```
TO GET.SET
PENUP
SETPOS [-15 0]
PENDOWN
FD :ORIG
PU
SETPOS [15 0]
PD
END
```

```
TO STUDENT.DRAW
PRINT [ ]
PRINT [TYPE F TO GO FORWARD, OR]
PRINT [S TO STOP.]
MAKE "ANS READCHAR
IF :ANS = "F [TAKE.A.STEP]
IF NOT (:ANS = "S) [STUDENT.DRAW]
END
```

```
TO TAKE.A.STEP
FD 5
MAKE "ST.ANS (:ST.ANS + 5)
END
```

```
TO EVALUATE
IF :ST.ANS > :ORIG [PRINT [YES, THAT IS LONGER]]
IF :ST.ANS = :ORIG [PRINT [THE 2 LINES ARE THE SAME!!!]]
IF :ST.ANS < :ORIG [PRINT [YOURS IS SHORTER]]
PRINT [ ]
END
```

```
TO CA
CLEARSCREEN
PENUP
RIGHT 55
SETPOS [-120 -75]
PENDOWN
END
```

```
TO CV
CLEARSCREEN
PENUP
BACK 79
PENDOWN
END
```

```
TO CL
CLEARSCREEN
PENUP
RIGHT 90
BACK 120
PENDOWN
END
```

PERIMETERS

TO EX1
PU
CS
SETPOS [-90 0]
PD
REPEAT 4 [FD 50 RT 90]
PU
SETPOS [0 0]
PD
END

TO EX2
CS
PU
SETPOS [-90 0]
PD
RT 30
REPEAT 3 [FD 50 RT 120]
PU
SETPOS [0 0]
PD
END

TO EX3
PU
CS
SETPOS [-90 0]
PD
REPEAT 5 [FD 30 RT 72]
PU
SETPOS [0 0]
PD
END

TO EX4
CS
PU
SETPOS [-90 0]
PD
REPEAT 6 [FD 15 RT 60]
PU
SETPOS [0 0]
PD
END

TO EX5
PU
CS
SETPOS [-90 0]
PD
REPEAT 2 [FD 30 RT 90 FD 50 RT 90]

PU
SETPOS [0 0]
PD
END

TO EX6
CS
PU
SETPOS [-90 0]
PD
REPEAT 4 [FD 30 RT 90 FD 30 RT 90 FD 30 LT 90]
PU
SETPOS [20 0]
PD
SETH 0
END

EST.DIST

TO TRY.ONE
WRAP
TARGET
ARCHER
SHOOT
AGAIN
END

TO TARGET
CLEARSCREEN
HIDETURTLE
MAKE "TARGET PICK.WHERE
NEIGHBORHOOD :TARGET
END

TO NEIGHBORHOOD :WHERE
PENUP
SETPOS :WHERE
PENDOWN
DOTS3
RT 90 FD 1 LT 90 DOTS3
LT 90 FD 2 RT 90 DOTS3
PENUP
END

TO DOTS3
FD 1 BK 2 FD 1
END

TO ARCHER
SETPOS PICK.WHERE
PENDOWN SHOWTURTLE
SETHEADING TOWARDS :TARGET
END

```
TO PICK.WHERE
OUTPUT LIST (137 - RANDOM 276) (117 - RANDOM 196)
END
```

```
TO SHOOT
TYPE [HOW FAR?\ \ ]
FD FIRST READLIST
END
```

```
TO AGAIN
TYPE [DO YOU WANT TO TRY ANOTHER? \ (Y\N\)\ ]
IF CHECK FIRST READLIST = "Y [CLEARTXT TRY.ONE] [TEXTSCREEN]
END
```

```
TO CHECK :IN
TEST OR :IN = "Y :IN = "N
IFFALSE [TYPE [Y OR N ONLY:\ ] OUTPUT CHECK FIRST RL]
IFTRUE [OUTPUT :IN]
END
```

EST.TURN

```
TO TRY.ONE
PENDOWN
WRAP
TARGET
ARCHER
SHOOT
AGAIN
END
```

```
TO TARGET
CLEARSCREEN HIDETURTLE
MAKE "TARGET PICK.WHERE
NEIGHBORHOOD :TARGET
END
```

```
TO PICK.WHERE
OUTPUT LIST (137 - RANDOM 276) (117 - RANDOM 196)
END
```

```
TO NEIGHBORHOOD :WHERE
PENUP
SETPOS :WHERE
PENDOWN
DOTS3
RT 90 FD 1 LT 90 DOTS3
LT 90 FD 2 RT 90 DOTS3
PENUP
END
```



```
TO DOTS3
FD 1 BK 2 FD 1
END
```

```
TO ARCHER
MAKE "ARCHER PICK.WHERE
SETPOS :ARCHER
PENDOWN SHOWTURTLE
END
```

```
TO SHOOT
TYPE [HOW MUCH TURN?\ \ ]
SETHEADING FIRST READLIST
FORWARD SQRT ((SQR XDIFF :TARGET :ARCHER) + (SQR YDIFF :TARGET :ARCHER))
END
```

```
TO SQR :NUM
OUTPUT :NUM * :NUM
END
```

```
TO XDIFF :P1 :P2
OUTPUT ((FIRST :P1) - (FIRST :P2))
END
```

```
TO YDIFF :P1 :P2
OUTPUT ((LAST :P1) - (LAST :P2))
END
```

```
TO AGAIN
TYPE [DO YOU WANT TO TRY ANOTHER? \ (Y\N)\ \ ]
IF CHECK FIRST READLIST = "Y [CLEARTEXT TRY.ONE] [TEXTSCREEN]
END
```

```
TO CHECK :IN
TEST OR :IN = "Y :IN = "N
IFFALSE [TYPE [Y OR N ONLY:\ ] OUTPUT CHECK FIRST READLIST]
IFTRUE [OUTPUT :IN]
END
```

AXES

```
TO AXES
CLEARSCREEN
FORWARD 240
RIGHT 90
FORWARD 280
HIDETURTLE
FENCE
END
```

SEQ.SERIES

TO ARITHMETIC

(LOCAL "FIRST "COMMON "N)

CLEARTEXT

PRINT [NOTE: ENTER ALL FRACTIONAL VALUES AS]

PRINT [DECIMALS.]

PRINT []

TYPE [FIRST TERM?\]

MAKE "FIRST FIRST READLIST

TYPE [COMMON DIFFERENCE?\]

MAKE "COMMON FIRST READLIST

TYPE [HOW MANY TERMS? \]

MAKE "N FIRST READLIST

PRINT []

ARITH.SEQ.LOOP :FIRST :COMMON 1 :N

PRINT []

PRINT [SEQUENCE SAMPLE COMPLETED]

END

TO ARITH.SEQ.LOOP :VALUE :TO.ADD :WHICH.TERM :HOW.MANY

(PRINT [TERM] (WORD "# :WHICH.TERM) :VALUE)

TEST :WHICH.TERM < :HOW.MANY

IFTRUE [ARITH.SEQ.LOOP (:VALUE + :TO.ADD) :TO.ADD (:WHICH.TERM + 1) :HOW.MANY]

END

TO GEOMETRIC

(LOCAL "FIRST "MULTIPLY "N)

CLEARTEXT

PRINT [NOTE: ENTER ALL FRACTIONAL VALUES AS]

PRINT [DECIMALS.]

PRINT []

TYPE [FIRST TERM?\]

MAKE "FIRST FIRST READLIST

TYPE [COMMON RATIO?\]

MAKE "MULTIPLY FIRST READLIST

TYPE [HOW MANY TERMS? \]

MAKE "N FIRST READLIST

PRINT []

GEO.SEQ.LOOP :FIRST :MULTIPLY 1 :N

PRINT []

PRINT [SEQUENCE SAMPLE COMPLETED]

END

TO GEO.SEQ.LOOP :VALUE :TO.MULTIPLY :WHICH.TERM :HOW.MANY

(PRINT [TERM] (WORD "# :WHICH.TERM) :VALUE)

TEST :WHICH.TERM < :HOW.MANY

IFTRUE [GEO.SEQ.LOOP (:VALUE * :TO.MULTIPLY)
:TO.MULTIPLY (:WHICH.TERM + 1) :HOW.MANY]

END

```

TO SEQUENCE
LOCAL "LAST
CLEARTEXT
PRINT [DEFINE THE NTH TERM \ (IN TERMS OF N\):\ ]
PRINT [ ]
TYPE [\ \ \ \ \ \ \ \- \->]
MAKE "TERM READLIST
PRINT [ ]
TYPE [HOW MANY TERMS TO DISPLAY?\ ]
MAKE "LAST FIRST READLIST
PRINT [ ]
PRINT [ ]
SEQUENCE.LOOP :TERM 1 :LAST
END

```

```

TO SEQUENCE.LOOP :TERM.FORMULA :WHICH :LENGTH
(PRINT [TERM] (WORD "# :WHICH) RUN :TERM.FORMULA)
TEST :WHICH < :LENGTH
IFTRUE [SEQUENCE.LOOP :TERM.FORMULA :WHICH + 1 :LENGTH]
END

```

```

TO SERIES
LOCAL "LAST
CLEARTEXT
PRINT [DEFINE THE NTH TERM \ (IN TERMS OF N\):\ ]
PRINT [ ]
TYPE [\ \ \ \ \ \ \ \- \->]
MAKE "TERM READLIST
PRINT [ ]
TYPE [HOW MANY TERMS TO CONSIDER?\ ]
MAKE "LAST FIRST READLIST
PRINT [ ]
PRINT [ ]
SERIES.LOOP :TERM 0 1 :LAST
END

```

```

TO SERIES.LOOP :TERM.FORMULA :TO.ADD :WHICH :HOW.MANY
(PRINT [SUM OF THE FIRST] (:WHICH - 1) "TERM\ (S\ ) :TO.ADD)
TEST :WHICH < (:HOW.MANY + 1)
IFTRUE [SERIES.LOOP :TERM.FORMULA ((RUN :TERM.FORMULA) + :TO.ADD)
(:WHICH + 1) :HOW.MANY]
END

```

```

TO N
OUTPUT :WHICH
END

```

LIMITS

```

TO FIGURE :NUMBER
CLEARSCREEN
NGON :NUMBER (360 / :NUMBER)
END

```

```
TO NGON :NUM.SIDES :SIDE.LENGTH
REPEAT :NUM.SIDES [FORWARD :SIDE.LENGTH RIGHT (360 / :NUM.SIDES)]
END
```

```
TO MAG.FIGURE :NUMBER
CLEARSCREEN WINDOW
NGON :NUMBER 50
WRAP
END
```

```
TO GET.READY
TEXTSCREEN
CLEARTEXT
SETCURSOR [0 10]
MAKE "DIST.LEFT 2
MOVE
END
```

```
TO MOVE
MAKE "DIST.LEFT (:DIST.LEFT / 2)
(PRINT [THE TURTLE IS NOW] :DIST.LEFT [METERS])
PRINT [\ \ \ \ \ \ FROM THE WALL.]
END
```

GRAPHING

```
TO SETUP
NORMAL.SPEED
USE WHITE
WINDOW
HIDETURTLE
DRAW.AXES
USE ORANGE
USE SQUARES
NOTRAIL
END
```

```
TO DRAW.AXES
CLEARSCREEN
AXIS 11
RIGHT 90
AXIS 12
RIGHT 90
AXIS 7
RIGHT 90
AXIS 12
RIGHT 90
END
```

```
TO AXIS :NUM.INTERVALS
PENDOWN
REPEAT :NUM.INTERVALS [FORWARD 10 TICK.MARK]
FORWARD 10
```

```
PENUP
SETPOS [0 0]
PENDOWN
END
```

```
TO TICK.MARK
RIGHT 90
FORWARD 3
BACK 6
FORWARD 3
LEFT 90
END
```

```
TO GRAPH
GET.RULE
PLOT.LEFTMOST.POINT
GRAPH.LOOP
PENUP
SETPOS [0 0]
END
```

```
TO GET.RULE
PRINT [GIVE THE FUNCTION RULE ( IN TERMS OF X )]
TYPE [Y = \ ]
MAKE "RULE READLIST
END
```

```
TO PLOT.LEFTMOST.POINT
PENUP
MAKE "X -13
SETPOS LIST ( 10 * :X ) ( 10 * RUN :RULE )
PENDOWN
RUN LIST :SHAPE
END
```

```
TO GRAPH.LOOP
MAKE "X (:X + :DELTA)
PLOT :X (RUN :RULE)
IF :X < 13 [GRAPH.LOOP]
END
```

```
TO X
OUTPUT :X
END
```

```
TO PLOT :XVAL :YVAL
RAWPLOT (10 * :XVAL) (10 * :YVAL)
END
```

```
TO RAWPLOT :XVAL :YVAL
IF :TRAIL = "Y [PENDOWN] [PENUP]
SETPOS LIST :XVAL :YVAL
RUN LIST :SHAPE
END
```

TO X'S
MAKE "SHAPE "AN.X
END

TO AN.X
LOCAL "HOLDPOS
MAKE "HOLDPOS POS
PENDOWN
SETHEADING 45 FD 3 BK 6 FD 3 LT 90 FD 3 BK 6 FD 3 RT 45
PENUP SETPOS :HOLDPOS
END

TO SQUARES
MAKE "SHAPE "SQUARE
END

TO SQUARE
LOCAL "HOLDPOS
MAKE "HOLDPOS POS
PENUP SETHEADING 45 BACK 1.4 LT 45
PENDOWN REPEAT 4 [FD 2 RT 90]
PENUP SETPOS :HOLDPOS
END

TO CROSSES
MAKE "SHAPE "CROSS
END

TO CROSS
LOCAL "HOLDPOS
MAKE "HOLDPOS POS
PENDOWN SETHEADING 0
FD 3 BK 6 FD 3 LT 90 FD 3 BK 6 FD 3 RT 90
PENUP SETPOS :HOLDPOS
END

TO POINTS
MAKE "SHAPE "POINT
END

TO POINT
PENDOWN DOT POS PENUP
END

TO TRAIL
MAKE "TRAIL "Y
END

TO NOTRAIL
MAKE "TRAIL "N
END

TO WHITE
SETPC 1
END

TO ORANGE
SETPC 4
END

TO BLUE
SETPC 5
END

TO CAREFULLY
MAKE "DELTA .1
END

TO NORMAL.SPEED
MAKE "DELTA .5
END

TO QUICKLY
MAKE "DELTA 2.
END

TO TIDY.UP
SETPC 1
PENDOWN
SHOWTURTLE
WRAP
CLEARSCREEN
CLEARTEXT
END

MOVE1

TO MOVE :DISTANCE :TIME
FULLSCREEN
FORWARD :DISTANCE
WAIT :TIME
MOVE :DISTANCE :TIME
END

MOVE2

TO MOVE :DISTANCE :TIME :ACC
FULLSCREEN
FORWARD :DISTANCE
WAIT :TIME
MOVE (:DISTANCE + :ACC) :TIME :ACC
END

MOVE3

```
TO MOVE :XDIST :YDIST :TIME
FULLSCREEN
SETX XCOR + :XDIST
SETY YCOR + :YDIST
WAIT :TIME
MOVE :XDIST :YDIST :TIME
END
```

MOVE4

```
TO MOVE :XDIST :YDIST :TIME
FULLSCREEN
SETPOS LIST (XCOR + :XDIST) (YCOR + :YDIST)
WAIT :TIME
MOVE :XDIST :YDIST :TIME
END
```

MOVE5

```
TO MOVE :XDIST :YDIST :TIME :XACC :YACC
FULLSCREEN
SETPOS LIST ( XCOR + :XDIST ) ( YCOR + :YDIST )
WAIT :TIME
MOVE ( :XDIST + :XACC ) ( :YDIST + :YACC ) :TIME :XACC :YACC
END
```

MOVE6

```
TO MOVE :DISTANCE :TIME
FULLSCREEN
IF KEYP [DO.SOMETHING.WITH RC]
FORWARD :DISTANCE
WAIT :TIME
MOVE :DISTANCE :TIME
END
```

```
TO DO.SOMETHING.WITH :KEY
IF EQUALP :KEY "F [MAKE "DISTANCE :DISTANCE + 1]
IF EQUALP :KEY "S [MAKE "DISTANCE :DISTANCE - 1]
END
```

BOUNCING

```
TO DROP
WRAP
SETUP
LOCAL "YVEL
MAKE "YVEL SPECS
SET.TURTLE
NAME 0 "BC
```

```

CLEARTEXT
FALL YCOR :YVEL
SPLITSSCREEN
PR (SE [THE BALL BOUNCED] :BC "TIMES.)
END

```

```

TO SETUP
CLEARTEXT
SPLITSSCREEN
HT
END

```

```

TO SPECS
PR [WHAT ACCELERATION?\ \ ]
NAME ( ( 0 - FIRST RL ) / 10 ) "YACC
PR [WHAT INITIAL HORIZONTAL VELOCITY?]
NAME FIRST RL "XVEL
PR [WHAT IS THE COEFFICIENT OF]
TYPE [ELASTICITY \ (0 TO 1\)?\ \ ]
NAME ( 0 - FIRST RL ) "ELAS
PR [WHAT INITIAL DOWNWARD VELOCITY?]
OP 0 - FIRST RL
END

```

```

TO SET.TURTLE
PU
MAKE "PENSTAT PEN
SETPOS [-140 -118]
SETH 90
PD
FD 280
PU
PR [FROM HOW HIGH SHOULD IT BE DROPPED?]
TYPE [( 0 TO 236 ) \ \ ]
SETPOS LIST -140 (-118 + FIRST RL)
FULLSCREEN
DOT LIST XCOR YCOR
SETPEN :PENSTAT
END

```

```

TO FALL :YPOS :YVEL
TEST (:YPOS + :YVEL + :YACC) < -118
IFF [SETPOS LIST SUM XCOR :XVEL (:YPOS + :YVEL + :YACC)]
IFF [PD DOT LIST XCOR YCOR SETPEN :PENSTAT]
IFF [FALL (:YPOS + :YVEL + :YACC) (:YVEL + :YACC)]
IFT [IF :YVEL < -.6 [SETPOS LIST SUM XCOR :XVEL (-236 - (:YPOS + :YVEL + :YACC))]]
IFT [IF :YVEL < -.6 [MAKE "BC :BC + 1 DOT LIST XCOR YCOR FALL YCOR :YVEL * :ELAS]]
END

```

```

TO LINE :COLOR
PD
SETPC :COLOR
END

```

TO DOTS :COLOR
PU
SETPC :COLOR
END

TO PAGE :COLOR
SETBG :COLOR
END

TO BLUE
OP 5
END

TO ORANGE
OP 4
END

TO PURPLE
OP 3
END

TO GREEN
OP 2
END

TO WHITE
OP 1
END

TO BLACK
OP 0
END

TO DRAW
END

TO ON
END

TO A
END

TO THE
END

TO BALL
END

TO TIDY.UP
PENDOWN
SETPC 1
SETBG 0
SHOWTURTLE
CLEARSCREEN

TEXTSCREEN
CLEARTEXT
END

BOUNCING2

TO TIDY.UP
PENDOWN
SHOWTURTLE
CLEARSCREEN
TEXTSCREEN
CLEARTEXT
END

TO FALL :YPOS :YVEL
TEST (:YPOS + :YVEL + :YACC) < -118
IFF [SETPOS LIST SUM XCOR :XVEL (:YPOS + :YVEL + :YACC)]
IFF [PD DOT LIST XCOR YCOR SETPEN :PENSTAT]
IFF [FALL (:YPOS + :YVEL + :YACC) (:YVEL + :YACC)]
IFT [IF :YVEL < -.6 [SETPOS LIST SUM XCOR :XVEL (-236 - (:YPOS + :YVEL + :YACC))]]
IFT [IF :YVEL < -.6 [MAKE "BC :BC + 1 DOT LIST XCOR YCOR FALL YCOR :YVEL * :ELAS]]
END

TO SET.TURTLE
PU
MAKE "PENSTAT PEN
SETPOS [-140 -118]
SETH 90
PD
FD 280
PU
PR [FROM HOW HIGH SHOULD IT BE DROPPED?]
TYPE [(0 TO 236) \ \]
SETPOS LIST -140 (-118 + FIRST RL)
FULLSCREEN
DOT LIST XCOR YCOR
SETPEN :PENSTAT
END

TO SPECS
PR [WHAT ACCELERATION?\ \]
NAME FIRST RL "YACC
MAKE "YACC -:YACC / 10
PR [WHAT INITIAL HORIZONTAL VELOCITY?]
NAME FIRST RL "XVEL
PR [WHAT IS THE COEFFICIENT OF]
TYPE [ELASTICITY \ (0 TO 1)\ ?\ \]
NAME (0 - FIRST RL) "ELAS
PR [WHAT INITIAL DOWNWARD VELOCITY?]
OP 0 - FIRST RL
END

```

TO SETUP
  CLEARTEXT
  SPLITSSCREEN
  HT
  END

```

```

TO DROP
  WRAP
  SETUP
  LOCAL "YVEL
  MAKE "YVEL SPECS
  SET.TURTLE
  NAME 0 "BC
  CLEARTEXT
  FALL YCOR :YVEL
  SPLITSSCREEN
  PR (SE [THE BALL BOUNCED] :BC "TIMES.)
  END

```

PLANT.GAME

```

TO INFO
  CLEARTEXT
  SETCURSOR [4 2]
  PR [THIS GAME IS SIMPLE TO PLAY:]
  PR [ ]
  PR [YOU THINK OF A PLANT, AND LOGO]
  PR [TRIES TO GUESS WHAT YOU HAVE CHOSEN]
  PR [BY ASKING YES\ -OR\ -NO QUESTIONS.]
  PR [ ]
  PR [THE LONGER YOU PLAY, THE BETTER THE]
  PR [COMPUTER WILL BECOME AT GUESSING!]
  PR [ ]
  PR [(TRANSLATED FROM A PROGRAM WRITTEN IN]
  PR [LISP BY BERNARD GREENBERG. \)]
  PR [ ]
  PR [(... AND ADAPTED BY RICK HUNYADY \)]
  REPEAT 3 [PR [ ]]
  PR [PRESS RETURN TO BEGIN]
  TYPE RC
  CLEARTEXT SETCURSOR [0 0]
  PLANT.GAME
  END

```

```

TO PLANT.GAME
  CLEARTEXT
  SETCURSOR [5 5]
  PR [ARE YOU STARTING A NEW GAME \ (Y OR N) ?]
  IF RW = "N [STARTTREE] [INIT]
  PLAY
  END

```

```

TO INIT
MAKE "TREE [[IS THIS A NONFLOWERING PLANT?] MOSS ROSE]
END

```

```

TO STARTTREE
PR [ ]
PR [TYPE IN THE NAME YOU USED TO SAVE]
PR [YOUR LAST GAME.]
LOAD WORD FIRST RL ".PL.DA
END

```

```

TO GETNEWSPECIE
PR [I GIVE UP. WHAT WAS YOUR PLANT?]
OP GET [C'MON, WHAT WAS IT REALLY?]
END

```

```

TO RECORD
PR [DO YOU WISH TO SAVE YOUR PLANT INFO?]
PR [ ]
TYPE [YES OR NO\ \ ]
TEST FIRST RL = "YES
IFF [ERNS STOP]
IFT [PR [ENTER YOUR NAME ( NO SPACES )]]
IFT [PR [IT MUST NOT HAVE BEEN USED ON THE DISK BEFORE.]]
IFT [SAVE WORD FIRST RL ".PL.DA]
UNBURY "PLANTS
ERALL
END

```

```

TO PLAY
PR [THINK OF A PLANT, ANY PLANT.]
PR "\
EXPLORE :TREE
IF ASK [DO YOU WANT TO PLAY AGAIN?] [PLAY] [RECORD]
END

```

```

TO EXPLORE :NODE
TEST WORDP :NODE
IFT [IF ASK (SE [IS IT] A.OR.AN :NODE "? ) [PR [I GUESSED IT!] STOP] [AI :NODE STOP]]
IFF [TEST ASK FIRST :NODE]
IFT [EXPLORE FIRST BF :NODE]
IFF [EXPLORE FIRST BF BF :NODE]
END

```

```

TO ASK :MESSAGE
PR :MESSAGE
LOCAL "ANS
MAKE "ANS RL
IF EMPTY :ANS [COMPLAIN OP ASK :MESSAGE]
IF NOT MEMBERP FIRST :ANS [YES NO] [COMPLAIN OP ASK :MESSAGE]
OP IF FIRST :ANS = "YES ["TRUE] ["FALSE]
END

```

```

TO COMPLAIN
PR [PLEASE ANSWER YES OR NO]
END

TO DOUBLE.CONSONANT? :SPECIEPART
IF AND LAST :SPECIEPART = "E LAST BL :SPECIEPART = LAST BL BL :SPECIEPART
    [OP (SE [DOES THIS PLANT] BL :SPECIEPART BF BF :DIF "?)]
OP (SE [DOES THIS PLANT] :SPECIEPART BF BF :DIF "?)
END

TO A.OR.AN :WORD
IF MEMBERP FIRST :WORD [A E I O U] [OP SE "AN :WORD]
OP SE "A :WORD
END

TO VERB.TEST :DIF
LOCAL "SPECIEPART
MAKE "SPECIEPART FIRST BF :DIF
IF LAST :SPECIEPART = "S [OP DOUBLE.CONSONANT? BL :SPECIEPART]
OP (SE [IS IT TRUE THAT] :DIF "?)
END

TO REPLACE :BIGLIST
IF :BIGLIST = :LOSER [OP :SUBLIST]
IF WORDP :BIGLIST [OP :BIGLIST]
OP (LIST FIRST :BIGLIST REPLACE FIRST BF :BIGLIST REPLACE LAST :BIGLIST)
END

TO GET :REPLY
LOCAL "RESPONSE
MAKE "RESPONSE RL
IF MEMBERP :RESPONSE [I DO IF GET DONT [ ]] [PR :REPLY GET :REPLY] [OP :RESPONSE]
END

TO GETDIF
(PR [TELL ME SOMETHING ABOUT] A.OR.AN :NEWSPECIE
    [THAT'S NOT TRUE ABOUT] A.OR.AN :LOSER)
OP GET [COULD I HAVE THAT ONE MORE TIME?]
END

TO NEWQUESTION :DIF
IF AND MEMBERP FIRST :DIF [A AN] FIRST BF :DIF = :NEWSPECIE
    [MAKE "DIF SE "IT BF BF :DIF]
IF FIRST BF :DIF = "IS [OP (SE [IS THIS PLANT] BF BF :DIF "?)]
IF FIRST BF :DIF = "HAS [OP (SE [DOES IT HAVE] BF BF :DIF "?)]
OP VERB.TEST :DIF
END

TO ALTER.NODE :QUESTION
LOCAL "SUBLIST
MAKE "SUBLIST (LIST :QUESTION :NEWSPECIE :LOSER)
MAKE "TREE REPLACE :TREE
END

```



```

TO AI :NODE
LOCAL "LOSER
LOCAL "NEWSPECIE
MAKE "LOSER :NODE
MAKE "NEWSPECIE LAST GETNEWSPECIE
ALTER.NODE NEWQUESTION GETDIF
END

```

```

TO RW
OP FIRST RL
END

```

```

MAKE "STARTUP [IF NOT KEYP [BURY "PLANTS INFO]]
PPROP "INIT "PROCPKG "PLANTS
PPROP "STARTTREE "PROCPKG "PLANTS
PPROP "GETNEWSPECIE "PROCPKG "PLANTS
PPROP "RECORD "PROCPKG "PLANTS
PPROP "INFO "PROCPKG "PLANTS
PPROP "PLAY "PROCPKG "PLANTS
PPROP "EXPLORE "PROCPKG "PLANTS
PPROP "ASK "PROCPKG "PLANTS
PPROP "COMPLAIN "PROCPKG "PLANTS
PPROP "DOUBLE.CONSONANT? "PROCPKG "PLANTS
PPROP "A.OR.AN "PROCPKG "PLANTS
PPROP "VERB.TEST "PROCPKG "PLANTS
PPROP "REPLACE "PROCPKG "PLANTS
PPROP "GET "PROCPKG "PLANTS
PPROP "GETDIF "PROCPKG "PLANTS
PPROP "NEWQUESTION "PROCPKG "PLANTS
PPROP "ALTER.NODE "PROCPKG "PLANTS
PPROP "AI "PROCPKG "PLANTS
PPROP "RW "PROCPKG "PLANTS
PPROP "PLANT.GAME "PROCPKG "PLANTS
PPROP "STARTUP "VALPKG "PLANTS

```

ELEMENT.GAME

```

TO COMPLAIN
PR [PLEASE ANSWER YES OR NO]
END

```

```

TO GETNEWATOM
PR [I GIVE UP. WHAT WAS YOUR ELEMENT?]
OP GET [C'MON, WHAT WAS IT REALLY?]
END

```

```

TO ELEMENT.GAME
CLEARTEXT
SETCURSOR [5 5]
PR [ARE YOU STARTING A NEW GAME \ (Y OR N) ?]
IF FIRST RL = "N [STARTTREE] [INIT]
PLAY
END

```

193

```

TO INIT
MAKE "TREE [[IS THIS ELEMENT A NONMETAL?] OXYGEN IRON]
END

```

```

TO STARTTREE
PR [ ]
PR [TYPE IN THE NAME YOU USED TO SAVE]
PR [YOUR LAST GAME.]
LOAD WORD FIRST RL ".EL.DA
END

```

```

TO INFO
CLEARTEXT
SETCURSOR [4 2]
PR [THIS GAME IS SIMPLE TO PLAY:]
PR [ ]
PR [YOU THINK OF AN ELEMENT, AND LOGO]
PR [TRIES TO GUESS WHAT YOU HAVE CHOSEN]
PR [BY ASKING YES\ -OR\ -NO QUESTIONS.]
PR [ ]
PR [THE LONGER YOU PLAY THE BETTER THE]
PR [COMPUTER WILL BECOME AT GUESSING!]
PR [ ]
PR [\ (TRANSLATED FROM A PROGRAM WRITTEN IN]
PR [LISP BY BERNARD GREENBERG.\)]
PR [ ]
PR [\ (... AND ADAPTED BY RICK HUNYADY\)]
REPEAT 3 [PR [ ]]
PR [PRESS RETURN TO BEGIN]
TYPE RC
CLEARTEXT SETCURSOR [0 0]
ELEMENT.GAME
END

```

```

TO RECORD
PR [DO YOU WISH TO SAVE YOUR ELEMENT INFO?]
PR [ ]
TYPE [YES OR NO\ \ ]
TEST FIRST RL = "YES
IFF [ERNS STOP]
IFT [PR [ENTER YOUR NAME ( NO SPACES )]]
IFT [PR [IT MUST NOT HAVE BEEN USED ON THE DISK BEFORE.]]
IFT [SAVE WORD FIRST RL ".EL.DA]
UNBURY "ELEMENTS
ERALL
END

```

```

TO PLAY
PR [THINK OF AN ELEMENT, ANY ELEMENT.]
PR "\
EXPLORE :TREE
IF ASK [DO YOU WANT TO PLAY AGAIN?] [PLAY] [RECORD]
END

```

```

TO EXPLORE :NODE
TEST WORDP :NODE
IFT [IF ASK (SE [IS IT] :NODE "?") [PR [I GUESSED IT!] STOP] [AI :NODE STOP]]
IFF [TEST ASK FIRST :NODE]
IFT [EXPLORE FIRST BF :NODE]
IFF [EXPLORE FIRST BF BF :NODE]
END

TO ASK :MESSAGE
PR :MESSAGE
LOCAL "ANS
MAKE "ANS RL
IF EMPTY :ANS [COMPLAIN OP ASK :MESSAGE]
IF NOT MEMBERP FIRST :ANS [YES NO] [COMPLAIN OP ASK :MESSAGE]
OP IF FIRST :ANS = "YES ["TRUE] ["FALSE]
END

TO GETDIF
(PR [TELL ME SOMETHING ABOUT] :NEWATOM [THAT'S NOT TRUE ABOUT] :LOSER)
OP GET [COULD I HAVE THAT ONE MORE TIME?]
END

TO DOUBLE.CONSONANT? :ATMPART
IF AND LAST :ATMPART = "E LAST BL :ATMPART = LAST BL BL :ATMPART
  (OP (SE [DOES THIS ELEMENT] BL :ATMPART BF BF :DIF "?))
OP (SE [DOES THIS ELEMENT] :ATMPART BF BF :DIF "?
END

TO VERB.TEST :DIF
LOCAL "ATMPART
MAKE "ATMPART FIRST BF :DIF
IF LAST :ATMPART = "S [OP DOUBLE.CONSONANT? BL :ATMPART]
OP (SE [IS IT SO THAT] :DIF "?
END

TO REPLACE :BIGLIST
IF :BIGLIST = :LOSER [OP :SUBLIST]
IF WORDP :BIGLIST [OP :BIGLIST]
OP (LIST FIRST :BIGLIST REPLACE FIRST BF :BIGLIST REPLACE LAST :BIGLIST)
END

TO GET :REPLY
LOCAL "RESPONSE
MAKE "RESPONSE RL
IF MEMBERP :RESPONSE [I DO IF GET DONT [ ]] [PR :REPLY GET :REPLY] [OP :RESPONSE]
END

TO NEWQUESTION :DIF
IF AND MEMBERP FIRST :DIF [A AN] FIRST BF :DIF = :NEWATOM
  [MAKE "DIF SE "IT BF BF :DIF]
IF FIRST BF :DIF = "IS [OP (SE [IS THIS ELEMENT] BF BF :DIF "?)]
IF FIRST BF :DIF = "HAS [OP (SE [DOES IT HAVE] BF BF :DIF "?)]
OP VERB.TEST :DIF
END

```

```
TO ALTER.NODE :QUESTION
LOCAL "SUBLIST
MAKE "SUBLIST (LIST :QUESTION :NEWATOM :LOSER)
MAKE "TREE REPLACE :TREE
END
```

```
TO AI :NODE
LOCAL "LOSER
LOCAL "NEWATOM
MAKE "LOSER :NODE
MAKE "NEWATOM LAST GETNEWATOM
ALTER.NODE NEWQUESTION GETDIF
END
```

```
MAKE "STARTUP [IF NOT KEYP [BURY "ELEMENTS INFO]]
PPROP "INIT "PROCPKG "ELEMENTS
PPROP "STARTTREE "PROCPKG "ELEMENTS
PPROP "GETNEWATOM "PROCPKG "ELEMENTS
PPROP "RECORD "PROCPKG "ELEMENTS
PPROP "INFO "PROCPKG "ELEMENTS
PPROP "PLAY "PROCPKG "ELEMENTS
PPROP "EXPLORE "PROCPKG "ELEMENTS
PPROP "ASK "PROCPKG "ELEMENTS
PPROP "COMPLAIN "PROCPKG "ELEMENTS
PPROP "DOUBLE.CONSONANT? "PROCPKG "ELEMENTS
PPROP "VERB.TEST "PROCPKG "ELEMENTS
PPROP "REPLACE "PROCPKG "ELEMENTS
PPROP "GET "PROCPKG "ELEMENTS
PPROP "GETDIF "PROCPKG "ELEMENTS
PPROP "NEWQUESTION "PROCPKG "ELEMENTS
PPROP "ALTER.NODE "PROCPKG "ELEMENTS
PPROP "AI "PROCPKG "ELEMENTS
PPROP "ELEMENT.GAME "PROCPKG "ELEMENTS
PPROP "STARTUP "VALPKG "ELEMENTS
```



LOGO EXCHANGE

The Magazine for Logo Activities Worldwide

Now published by the International Council for Computers in Education, *Logo Exchange* is an international resource for Logo educators.

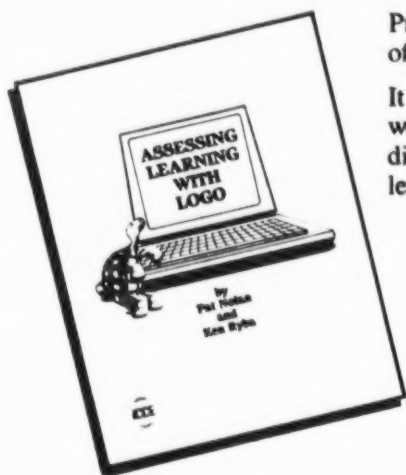
Nine times per year, *Logo Exchange* brings you exciting ideas from top Logo practitioners throughout the world.

Each issue includes:

- Columns to help teachers and parents develop activities
- Updates on research and technology
- Information on activities, meetings and conferences
- Reports from international correspondents
- Book reviews
- Product reviews
- Critiques
- Articles written by classroom teachers and parents

Logo Exchange is published by the International Council for Computers in Education (ICCE), the leading U.S. and international professional organization for computer educators. Supported by more than 50 organizations of computer-using educators from all over the world, ICCE is dedicated to the improvement of education through technology.

Published monthly September through May. Membership rates are \$24.95 per year. To order, contact ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403. Ph: (503) 686-4414.



Assessing Learning With Logo

Presents the method for *Assessing Learning With Logo* at the levels of basic Turtle commands, repeats and procedures.

It contains all the necessary materials—checklists, assessment worksheets and activities—for developing coding, exploration, prediction, analysis and planning, creativity and debugging at each level of learning Logo.

The methods and activities have been especially designed to highlight the role of the educator as a “facilitator of learning.” In this role educators guide students to reflect on their own thinking as they come into contact with powerful ideas at the beginning levels of Logo.

Single copies are \$12.50 plus \$2.50 shipping. Call now for a free catalog of ICCE publications.

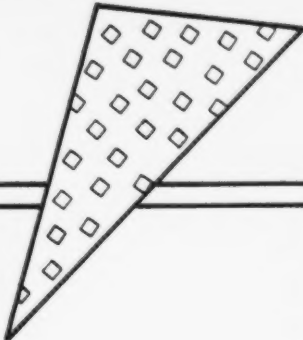


ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403. Ph: 503/686-4414.

Join the Leading Professional Organization for Computer Educators!

Whether you are an administrator, curriculum specialist, classroom teacher, media specialist or special educator, you will benefit from membership in ICCE.

ICCE is the leading U.S. and international professional organization for computer educators. It is non-profit, supported by more than 50 organizations of computer-using educators worldwide.



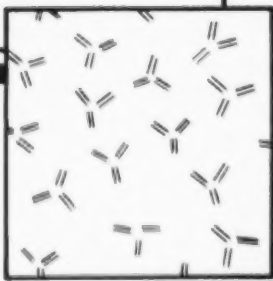
Membership in ICCE includes a subscription to *The Computing Teacher* journal. The journal has long been respected as an important source of information for computer educators, providing accurate, responsible and innovative information. Nine times per year, its articles, department and reviews keep you in touch with the constant changes taking place in educational technology.

You will also receive discounts on all ICCE publications and SIG memberships, and information on ICCE committee activities.

As educational technology continues to grow and change, look to ICCE and *The Computing Teacher* journal to keep you on the leading edge.

Join today!

ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403.
Ph: (503) 686-4414.



ISBN: 0-924667-41-9